LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO

SOLO PROGRAMADORES

ÑO VI. 2ª ÉPOCA NÚMERO 68 UNA PUBLICACIÓN DE: REVISTAS PROFESIONALES S.L. 975 Ptas. • 995 Escudos portugueses • 5,86 € (IVA incluido)

WINDOWS 2000 Server

ENTORNOS DE DESARROLLO
C++ Builder 5

TELECOMUNICACIONES
Curso de WAP

VISUAL BASIC
Programación orientada a objetos

DELPHI
Programación OpenGL

PROGRAMACIÓN WEB
Aplicaciones de Bases de Datos en Java

BASES DE DATOS
Aplicaciones de BBDD en Delphi

8 413042 303299 00068

CONTENIDO CD

Gran exclusiva: BORLAND C++BUILDER 5

● Borland JBuilder 3.5 Enterprise ● VisiBroker 4.0 for Java and C++ ● XML Junction 7.0 ● Relayfax 2.2.00 ● CodeGuide 2.5 ● KAWA 4.0 ● TiffDLL v3.17 ● ClonecCD 2.1.1.1 ● WinZip 8.0

PARA NO QUEDARSE HELADO CUANDO HABLEN DE LINUX

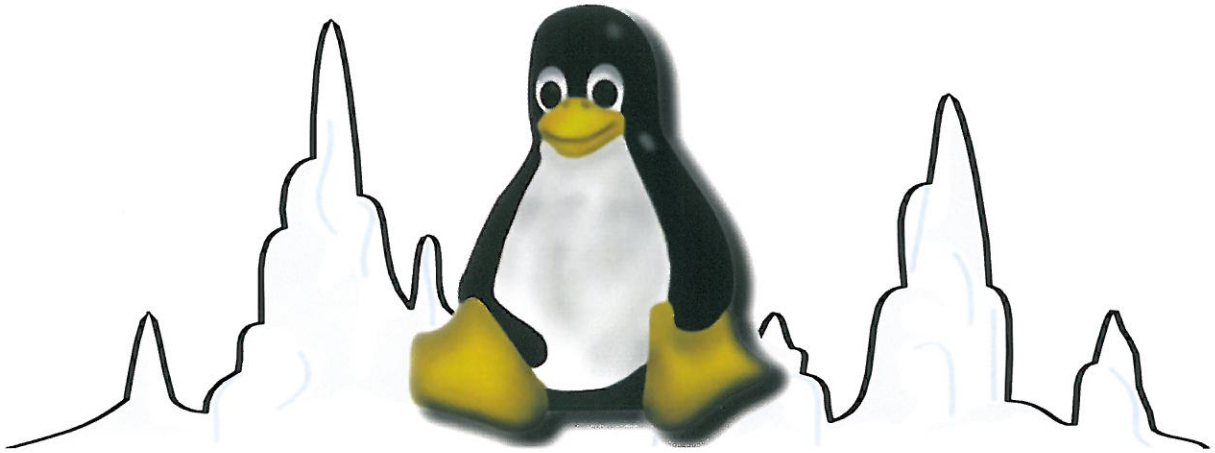
CONTENIDO CD

IMPRESCINDIBLES
BeOs-5.0 Personal edition • Cdrecord-1.8
Gimp 1.1.20 • Kernel 2.3.99pre5
OpenGui 2.17

DESTACADOS
Gnome 1.1.x • KDE2. Kdestudio
Lyx-1.1.4fix3 • MAM. Qt-2.1

INCLUYE:

ENTREVISTA con Miguel de Icaza
X-WINDOW: ICEWM. Ligero, rápido y funcional
LINUX-EXPO: Todas las novedades y la opinión de los expositores
UTILIDADES: Edición de documentos en LaTex
HERRAMIENTAS: Diseño fotográfico con GIMP
MULTIMEDIA: MP3 en Linux. Streaming
PROGRAMACIÓN GRÁFICA: Programación multimedia con SDL
SOFTWARE: Ofimática Linux. Agendas, correo y suites



SÓLO PROGRAMADORES

Número 68
SÓLO PROGRAMADORES
es una publicación de
REVISTAS PROFESIONALES S.L.

Editor
Agustín G. Buelta
Coordinador Técnico
Eduardo De Riquer Frutos
Coordinador de Redacción
Francisco Pino García
Edición
Gema Prados Mateos
Colaboradores
Constantino Sánchez, Juan Luis Ceada, Javier Sanz, Adolfo Aladro, Enrique de la Lastra, Marc Vaquer, Jordi Agost, Vicente A. Sánchez Werner
Imagen de portada
José Antonio Caguigal
Maquetación y Tratamiento de Imagen
Paco Risco
Consultas técnicas
atecnica@virtualsw.es

Asesoría de Publicidad
Carmina Ferrer
Tel.: (91) 304 78 46
Mariano Sánchez (Barcelona)
Tel.: (93) 322 12 38

Suscripciones
Rosa Tabares
Tel. (91) 304 87 64 Fax: (91) 327 13 03

Impresión
Printone, S.A
Distribución
Motorpress Ibérica

GRUPO MPI Motorpress-Ibérica

La revista Sólo Programadores no tiene por qué estar de acuerdo con las opiniones escritas por sus colaboradores en los artículos firmados. El editor prohibe expresamente la reproducción total o parcial de los contenidos de la revista sin su autorización escrita.

Depósito legal: M-26827-1994
ISSN: 1134-4792
PRINTED IN SPAIN
COPYRIGHT 31-8-2000
Precio en Canarias, Ceuta y Melilla: 938 ptas. sin I.V.A.
Con sobretasa aérea: 975 ptas.

AEPP
Asociación Española de Editoriales de Publicaciones Periódicas

EDITORIAL

Paisaje para una guerra

En estos momentos se está produciendo una guerra abierta, empleando todas las armas imaginables para conseguir el dominio en el campo de los sistemas operativos. Partimos de dos hechos irrefutables: en el apartado de los sistemas de sobremesa, el rey incontestable es *Microsoft Windows,* y entre los grandes sistemas domina *Unix*, aunque tiene muchos, digamos, dialectos. Y como es lógico en estos tiempos, todo esta cambiando rápidamente, por lo que resulta difícil valorar la situación real.

Por un lado *Linux*, la versión *Open Source* de *Unix*, está ganando adeptos a gran velocidad. Tanta, que por unidades instaladas, se sitúa ya como el segundo sistema operativo. El gran argumento que apoya este impresionante crecimiento es la libertad que da al usuario. Por decirlo en pocas palabras, *Linux* proporciona al usuario el código del sistema operativo para que pueda mejorarlo y resolver los problemas que se le presenten sin depender de nadie, mientras que con los sistemas propietarios, el usuario está en manos de sus proveedores. Para colmo de males -piensan algunos- *Linux* es un sistema operativo escalable, que puede usarse, tanto encapsulado en pequeños dispositivos, como para gobernar grandes sistemas servidores.

Esta realidad esta amargando la existencia, tanto a *Microsoft,* como a *Sun*, hasta ahora los líderes en sistemas operativos propietarios. Y las reacciones están llegando ya a la calle.

La respuesta de *Microsoft* a esta situación, esta siendo poco menos que espectacular. Por un lado, ha lanzado recientemente las dos primeras versiones de *Windows* para la empresa: *Windows 2000 Professinal* y *Windows 2000 Server*, y acaba de anunciar la *segunda beta* de *Windows Data Server*, que es la versión enfocada a los grandes centros de datos que puede soportar 32 *CPUs* en *cluster,* y tiene capacidad para gestionar 64 *GB* de memoria. Y por la parte del consumo, se está distribuyendo ya entre los grandes clientes la *release candidate* de *Windows Me*, con lo cual cabe esperar que su lanzamiento será para el verano.

En *Solo Programadores,* pensamos que debemos estar atentos a estos cambios, y ofrecer artículos de análisis como el que este mes ofrecemos en portada. Una acertada mezcla de análisis y herramientas de trabajo, hará que nuestros lectores sigan siendo fieles en el quiosco. En espera de sus comentarios, hasta el mes próximo.

SUMARIO

68

SÓLO PROGRAMADORES

6 Noticias

En esta sección encontrarás las novedades más interesantes y las noticias más importantes del mes. Un recorrido por la actualidad de la programación y la informática.

9 Contenido del CD-ROM

Seguro que te va a encantar: *Borland C++Builder, Borland JBuilder 3.5, CofeeCup HTML 8.2,* y *Jcreator 1.11.* Todo ello acompañado por las herramientas más útiles y las actualizaciones más interesantes: entornos de programación, lenguajres, redes... y por supuesto, los ejemplos, listados y códigos fuentes de los artículos de la revista.

22 Telecomunicaciones

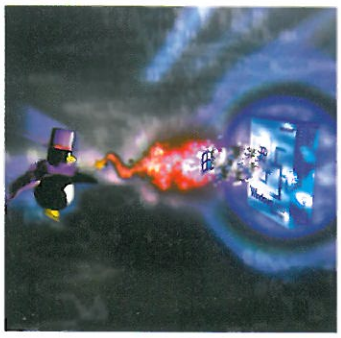
CURSO DE PROGRAMACIÓN CON WAPP (II)

En los números anteriores dedicamos estas páginas a conocer los conceptos básicos de la programación *WAP,* para adentrarnos más tarde en los lenguajes *WML* utilizados en los dispositivos de telefonía móvil. Ahora nos ocuparemos de ampliar y optimar las prestaciones de los móviles *WAP* mediante el uso de las variables. En este artículo aprenderemos a definir y nombrar variables, su validación, asignación de valores y sustitución, así como a utilizar el editor de *Toolkit.*

12 Portada Sistema Operativo

WINDOWS 200 SERVER CONTRA LINUX

Microsoft no está dispuesta a perder terreno en el campo de batalla de los sistemas operativos. Con la salida al mercado de *Windows 2000 Server* pretende afianzar su posición de dominancia, y hacer frente al envite de *Linux.* ¿Cuáles son las armas de cada contendiente en esta pugna? En este artículo vamos a comparar las características de *Windows,* con sus nuevas aportaciones, con las de su rival más cercano. Para ello, tomaremos la edición *Server* de *Windows* y la distribución *Mandrake 7.0-2* de *Linux.* Tendremos ocasión de comprobar cómo se vigilan de cerca los dos sistemas, y hasta qué punto las diferencias se están reduciendo



42 Entornos de Desarrollo

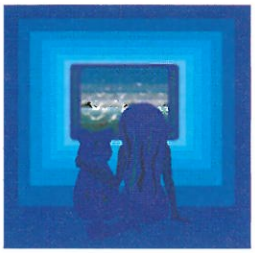
C++BUILDER 5

Con la presentación oficial de C++ Builder 5, Borland continúa el desarrollo de su mejor herramienta de programación para *C* y *C++* en esta quinta versión, disponible en sus variantes *Standard, Professional* y *Enterprise,* además de la edición gratuita *Compiler 5.5.* En este artículo se presentarán las novedades incluidas en esta versión. Hechas las primeras pruebas de evaluación, *C++Builder 5* se configura como una de las mejores herramientas de desarrollo, y se presenta como una alternativa muy seria a *Visual C++.*

28 Delphi

PROGRAMACIÓN OPENGL EN DELPHI (III)

Continuamos con la serie dedicada a la inserción de imágenes 3D mediante el estándar *Open Gl.* Tras estudiar las funciones de manipulación mediante el cálculo matricial, en esta ocasión vamos a aprender la manera de optimar nuestros programas, para que la visualización de los objetos sea más eficiente, y a crear objetos complejos a partir de otros más simples: dibujo de cuádricas, listas de visualización para mejorar la velocidad de dibujo, y ejemplos.



36 Visual Basic

PROGRAMACIÓN ORIENTADA A OBJETOS (III)

En el artículo anterior comenzamos a introducirnos en la programación de objetos en *Visual Basic,* estudiando sus características a través de las clases. Tras aprender a obtener información de las clases, para conocer las propiedades de los objetos, y a definir sus propiedades, en este número vamos a estudiar con más detenimiento las características de las clases de *Visual Basic 6,* para centrarnos más en el propio objeto.

50 Bases de datos

APLICACIÓN DE BASES DE DATOS CON DELPHI 5 (V)

Ha llegado el momento de entrar de lleno en la prática de la aplicación desarrollada en artículos anteriores. En este número vamos a modificar la aplicación para hacerla funcionar con diferentes gestores de bases de datos. Como anunciamos, lo primero que haremos es aprender a conectar nuestras aplicaciones *Delphi* con cualquier sistema gestor de bases de datos, centrántonos en *Interbase* y *Oracle.*

56 Entornos de Desarrollo

ENTORNO RAD JBUILDER 3.5

Borland presenta un nuevo y completo entorno de desarrollo rápido *Java* para diferentes plataformas, destinado a mejorar y ampliar las posibilidades de la versión 3.

64. Entornos de 64 Desarrollo

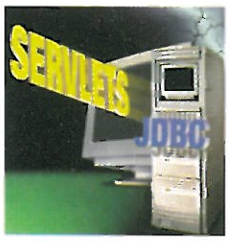
VISUAL C++ y MFC (y V)

Terminamos esta serie analizando las características de las cajas o cuadros de diálogo de *Windows* y los controles más comunes, para alcanzar una completa interacción con nuestros programas.

72 Bases de Datos

APLICACIONES WEB CON ACCESO A BBDD BASADAS EN JAVA (y VI)

Último artículo de la serie, dedicado a la descripción de las mejoras de rendimiento más interesantes para el acceso a base de datos de la nueva versión de la *API JDBC,* incluida en *JDK 1.2,* y que están destinadas a facilitar la labor de los desarrolladores de aplicaciones. Sin embargo, no debemos olvidar que algunas de estas características dependen del controlador *JDBC* utilizado.



78 Libros

En estas páginas encontraréis una selección de las novedades editoriales dedicadas a la promación más interesantes. Como de costumbre, cada reseña va acompañada de una ficha con información de precio, idioma, editorial, etcétera.

78 Ayuda al Lector

Dedicamos este espacio a contestar vuestras preguntas y resolver cualquier duda que queráis plantear. No dudéis en mandarnos vuestras consultas a la dirección **solop@virtualsw.es.** Como siempre, estaremos encantados de atenderos.

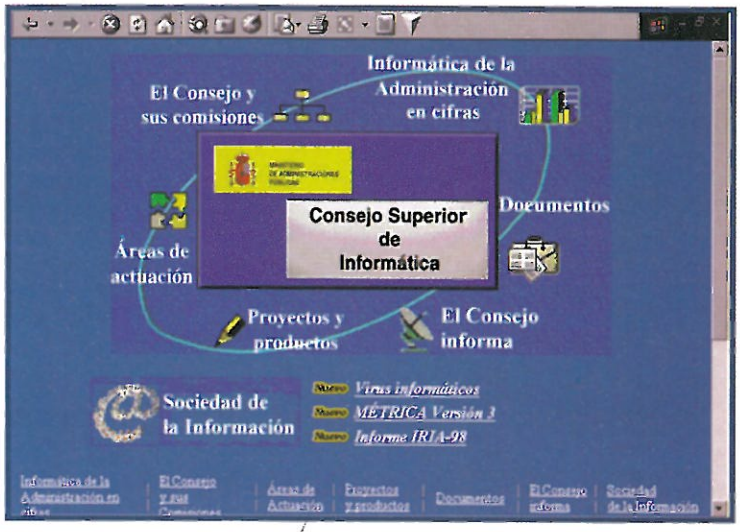
RED DE ALERTA TEMPRANA ANTI-VIRUS

El *Ministro de Administraciones Públicas*, Jesús posada, ha anunciado la creación de una *Red de Alerta Temprana* para prevenir los efectos de virus extendidos por *Internet*. La decisión, ha sido una de las primeras consecuencias de la aparición del virus *I love you*.

Recordemos que este gusano ha hecho sonar las alarmas de los sistemas informáticos de todo el mundo, llegando a colapsar al mismísimo Pentágono, aunque sus efectos en España no parecen haber sido de la misma magnitud.

A pesar de que cada día aparecen decenas de virus nuevos, este *bug* ha sorprendido por su destructividad y rapidez de propagación: en sólo 4 horas, ya se había extendido en 20 países. El virus, escrito en *VBScript*, se extiende por el correo electrónico e *IRC*, en ordenadores con *WSH*. Se descarga en la máquina en forma de fichero adjunto, con la extensión *.vbs*, y una vez instalado en el sistema, modifica el registro de *Windows*, y se reenvía a cualquier nuevo destinatario de correo. Su efecto más dañino radica en la corrupción de los archivos *.jpg, .jpeg, .mp3, .mp2, .vbs, .vbe, .js, .jse, .css, .wsh, .sct*, y *.hta*.

Además, en los últimos días, han aparecido nuevas variantes del *bug*, que pueden ser aún más dañinas. Esta circunstancia, deja en evidencia las limitaciones de seguridad de la *Red*, probando la facilidad con que un delincuente puede colapsar las instituciones públicas y ocasionar pérdidas multimillonarias en la empresa.



SCO SE UNE AL ESFUERZO DE HOMOLOGACIÓN EN LINUX

Santa Cruz Operation (SCO), ha anunciado su entrada como socio fundador en *FSC* (*Grupo de Estándares Libres*), un grupo de compañías que pretende establecer la *Base Estándar de Linux (LSB)*.

Según Mike Orr, presidente de *Tarantella (SCO)*, "La creación de un único estándar *Linux*, nos podrá ayudar a aumentar la compatibilidad de *Linux* con las plataformas de nuestros sistemas operativos. Desde un punto de vista más amplio, los *ISVs* se beneficiarán del estándar único, lo que incrementará el número de aplicaciones disponibles para plataformas *Linux* conformes con *LSB*, fortaleciendo a *UNIX* y *Linux* frente a *Intel*, e incrementando el valor de estas plataformas para el usuario."

FSG es una organización sin ánimo de lucro creada para acelerar la aceptación de la tecnología de fuentes abiertas mediante homologaciones de interoperabilidad, abarcando *Linux Standard Base* y *Linux Internationalization Initiative*. Entre los miembros fundadores de *FSG* están *IBM, LinuxCare, Sun, Red Hat, VA Linux, Caldera Systems, TurboLinux, SuSE y otros.*

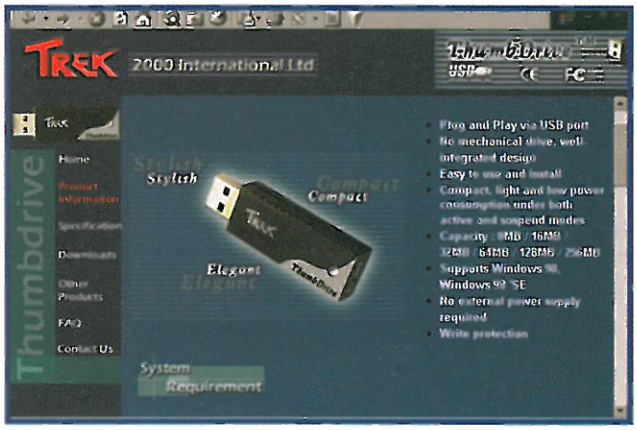
ACUERDO ENTRE MICROSOFT Y ANTENA 3 PARA WINDOWS MEDIA PLAYER

Antena 3i, filial *de Antena 3*, proveerá de contenidos a la versión española de *Media Guide*, la guía de recursos de audio y vídeo ofrecida a los clientes de *Microsoft* junto al *nuevo Windows Media Player.*

Media Guide en español, es una herramienta diseñada para dar acceso directo a contenidos *Web* de información y entretenimiento con imagen y sonido, de la que ya se han distribuido 100 millones de licencias en todo el mundo.

NUEVO SISTEMA DE ALMACENAMIENTO LIGERO

La empresa americana *Trek Technologies* ha presentado *Thumbdrive*, un nuevo dispositivo de almacenamiento ligero de un peso aproximado de 30 gramos y un tamaño de 10 X 2 centímetros, con una capacidad máxima de 256 *MB*.



Este minúsculo disco duro se conecta a un puerto *USB*, facilitando así la transferencia de archivos entre varios equipos. Por el momento, el dispositivo sólo está disponible en el mercado estadounidense, y se comercializa con diferente capacidad, desde los 8 a lo 256 *MB*. *Thumbdrive* es *Plug and Play*, por lo que no necesita ningún tipo de software de instalación, siendo compatible con *Windows 98*.

PARCHE DE SEGURIDAD PARA WINDOWS

Microsoft ha presentado un parche para solucionar un fallo de seguridad encontrado en su sistema operativo *Windows*, y que puede afectar el rendimiento de páginas *Web*. El origen del problema, está en la incapacidad de *Windows* para ensamblar correctamente algunos fragmentos *IP*, con lo que se puede producir el colapso de un ordenador ante un ataque de tipo *Ddos*.

Este parche solucionará el conflicto causado por envío de fragmentos *IP* con información incorrecta o corrupta, de manera que durante el tiempo de envío, el rendimiento de la máquina se ve mermado considerablemente, aunque la información crítica no esté en peligro.

Esta circunstancia ha sido aprovechada maliciosamente por los *Hackers*, que han llegado a colapsar algunas de las páginas *Web* más importantes, mediante el bombardeo de mensajes de correo electrónico, o solicitud de información a las páginas, para que los servidores queden bloqueados por sobrecarga.

ÉXITO DE JAVA EXPO 2000

Java Expo 2000 ha cerrado su tercera edición rodeada de un ambiente general de satisfacción. *Sun Microsistems* ha visto sus expectativas cumplidas, con un considerable aumento de público y expositores, mientras que estos últimos han tenido una ocasión única para estar al tanto de las últimas novedades producidas en un sector tan dinámico y creciente como es el Java.

En esta tercera edición, la feria se ha dirigido especialmente a concienciar a los empresarios de la necesidad de evolucionar con éxito hacia la *Red* sus negocios, aprovechando todas las ventajas del fenómeno *.com*.

Además de la propia feria, ha tenido una particular buena acogida las sesiones paralelas y *tracks* técnicos, donde más de 80 ponentes han sacado a la palestra los aspectos más interesantes y reveladores de las tecnologías *.com*.

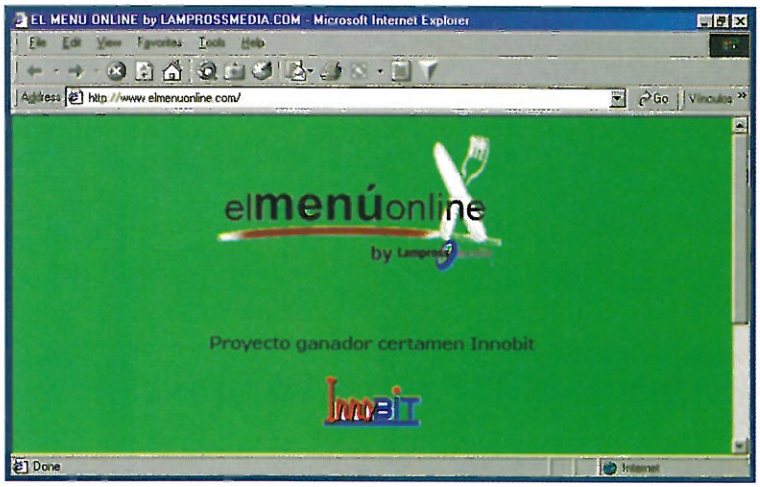
LAMPROSS MEDIA, GANADORA DEL CONCURSO DE IDEAS BIT 2000

La compañía *Lampross Media* ha sido la ganadora del concurso de ideas del certamen *Innobit*, celebrado en feria *BIT 2000* de Barcelona. El concurso ha estado patrocinado por *IBM, Terra y La Caixa*, quienes también han dotado el premio, consistente en una cantidad de dinero, y un ordenador portátil *IBM* de gama alta.

El proyecto ganador ha sido *"El Menú online"*, consiste en ofrecer a los establecimientos de hostelería la posibilidad de utilizar un servicio de distribución de menú diario a sus clientes, a través del correo electrónico.

"Lampross Media pretende de este modo unir la nueva tecnología a la más antigua tradición, transformando las clásicas pizarras de los restaurantes, en menús electrónicos, y acercar así las ventajas de la red a uno de los gremios más tradicionales que hasta el momento se mantenía un poco alejado de *Internet"*.

Innobit, dentro del salón *BiT* de barcelona, pretende ser un espacio para los *start-up's* de empresas y personas con proyectos innovadores en el campo de las *TI,* y ofrece la oportunidad de contactar con inversores que permitan desarrollar e implementar los proyectos.



APPLE RETRASA EL MAC OS X HASTA EL MES DE ENERO

Apple no presentará al mercado su versión final de sistema operativo de última generación hasta enero de 2001, lo que supone un retraso de 4 meses sobre la fecha anunciada anteriormente. La empresa ha justificado este retraso por la necesidad de dar más tiempo a los desarrolladores para terminar el producto, aunque para este verano ya estará disponible la versión *beta*.

Phill Schiller, Vicepresidente de Producto de Apple, ha manifestado al respecto que "Lo que antes estábamos llamando la versión final, ahora lo llamamos una versión *beta*. La verdad es que estamos lanzando el mismo software que estaba previsto, en el tiempo que estaba previsto, pero cambiando el nombre. Los clientes nos han pedido que haya una beta pública, con lo que en lugar de llamarlo versión 1.0, lo llamaremos *beta*".

A pesar de todo, *Mac OS X* ha recibido el respaldo de más de 200 firmas mundiales de desarrollo asistentes a la *Conferencia Mundial de Desarrolladores Apple (WWDC),* y empresas como *Adobe, Microsoft, IBM, HP, Canon, Macromedia, Epson, Avid, Digidesign, Disney Interactive, Palm* y *Sun Microsystems,* han manifestado su compromiso de lanzar al mercado productos para *Mac OS X.*

ENTORNOS DE PROGRAMACIÓN

ENTORNOS DE PROGRAMACIÓN

BORLAND C++ BUILDER 5 ENTERPRISE

Potente entorno de desarrollo de *C++ de Borland*. Puede generar rápidamente aplicaciones distribuidas y para *Internet*. Incluye un asistente de creación de sistemas extensibles capaces de soportar funciones de comercio electrónico utilizando *HTML* 4 y *XML*.

BORLAND JBUILDER 3.5 ENTERPRISE

Colección de herramientas visuales de desarrollo. Permiten construir aplicaciones *Pure JavaTM, applets, JSP/Servlets, JavaBeansTM, EJB* y aplicaciones distribuidas *CORBA* para *Java 2*.

GWD TEXT EDITOR 3.0

Editor de texto potente y fácil de usar que incluye un corrector ortográfico con soporte para 10 idiomas, una opción de auto corrección, un teclado configurable (con emulaciones de *QEDIT, Word y WordStar*); *syntax highlighting*, macros compatibles con *C*, soporte para código *C/C++* y los formatos de texto *DOS, UNIX y Mac*.

TEXTPAD 4.2 REV.2

Potente editor de texto para programadores compatible con *Java Development Toolkit*. Ofrece barras de herramientas personalizables, múltiples áreas de trabajo, soporte para ficheros *Unicode* y *UTF-8*, un asistente para crear clases de documentos, etcétera.

VISIBROKER 4.0 FOR JAVA AND C++

Completa solución *CORBA 2.3 Object Request Broker (ORB)* que soporta el desarrollo, comercialización y gestión de aplicaciones distribuidas de objetos a través de varias plataformas *hardware* y sistemas operativos.

XML JUNCTION 7.0

Herramienta para comercio electrónico y expertos de integración de aplicaciones. Automatiza la creación de *DTDs*, esquemas y documentos *XML* desde cualquier aplicación y formato de datos incluyendo *EDI, SQL, Legacy, HL7, SAP*, etcétera.

LENGUAJE

VISUAL BASIC

BESTOFWARE SMARTUI 1.1.104

Es un control *ActiveX* sin ventanas que da a las aplicaciones la apariencia de *Windows 2000*, *Office 2000* y otras interfaces de usuario. Un mismo control permite crear barras de menú, de herramientas, de estado, listas de opciones y propiedades, y otros *SmartObjects*.

ENTISOFT TOOLS 2.0

Se trata de una recopilación de 1.000 rutinas que abarcan numerosas categorías y funciones, como manipulación de cadenas, funciones matemáticas, operaciones con ficheros, estructuras, conversión y comparaciones de datos, etcétera.

VB CODE PRINT 6.11

Programa de ayuda que mejora las funciones de impresión de código fuente de *Visual Basic 6*. Permite imprimir cualquier parte del código fuente de un proyecto desde el propio entorno de desarrollo de *Visual Basic*.

VBEXPRESS LITE 2.0

Colección de utilidades para *Visual Basic*. Incluye una librería de código verificado y 16 utilidades orientadas a automatizar tareas repetitivas como la documentación de código, lo que permite al programador centrarse en los aspectos fundamentales de un desarrollo.

JAVA

CODEGUIDE 2.5

Es un entorno inteligente de desarrollo en *Java* que cuenta con un innovador sistema de análisis del código fuente, que permite compilar, probar y depurar los programas con unas pocas selecciones de ratón. Analiza el código de las aplicaciones según está siendo escrito y muestra instantáneamente los errores.

FORMULA ONE FOR JAVA 7.0.1

Aplicación de hoja de cálculo basada en *Java* que puede ser usada para añadir hojas de cálculo a páginas *HTML* y aplicaciones *Java*, aunque también funciona como un programa independiente. Puede ser usada en un servidor para que los *Webmasters* y desarrolladores *Java* recuperen, manipulen y den formato a datos de bases de datos y clientes conectados a una red, *Internet* o una *intranet*.

JCREATOR 1.11

Potente editor para tecnología *Java* que Ofrece funciones de gestión de proyectos, plantillas de proyect
múltiples compiladores, asi

números de línea y una interfaz de usuario totalmente personalizable.

KAWA 4.0

Entorno de desarrollo integrado para aplicaciones *Java*. La arquitectura basada en plugins Java ofrece una completa colección de *APIs* para su integración con otras herramientas y aplicaciones. Incluye un gestor visual de proyectos, un editor de clases y visores de clases y código fuente. Soporta *JavaBeans, XML* y *JPDA*.

HTML

COFFEECUP HTML EDITOR 8.2

Excelente editor *HTML* que ofrece numerosas ayudas y elementos para incluir en las páginas *Web*. Incluye *Expresso FTP* (que realiza transferencias de ficheros), un navegador interno para probar las páginas, una utilidad para previsualizar las imágenes, un sistema de ayuda en línea y un fichero de referencias de tags *HTML 4.0*

HOTDOG PROFESSIONAL 6.0

Herramienta de autor *HTML* para editar texto nativo. Incluye herramientas para editar ficheros remotos, añadir *plugins, GIFs* animados, crear botones, mapas de imágenes, animaciones y efectos de texto basados en *Java*, etcétera. y cuenta con un gestor de sitios. Soporta los lenguajes *HTML, DHTML, CSS, ASP, VBScript y JavaScript.*

IMS WEB ENGINE 1.77

Herramienta de autor que soporta *DHTML*. Su utilización es sencilla, ya que se apoya en el procedimiento de arrastrar y soltar para crear animaciones y secuencias de movimientos que siguen un orden jerárquico. Incluye un completo editor que maneja ficheros en formato *HTML, TXT y RTF*, y un editor de polígonos y vectores para crear botones y gráficos con mapas de textura.

XMLWRITER 1.21

Editor de validación *XML* que cuenta con una interfaz de usuario muy intuitiva que permite dar formato al texto *XML* utilizando hojas de estilo *XSL* o *CSS* y ofrece funciones avanzadas de edición. Soporta *CSS, DTD, HTML, XML* y ficheros de texto *XSL*.

OTROS

ACTIVEX INSTRUMENTATION LIBRARY 2.15

Colección de 14 componentes de interfaz. Entre otros controles incluye un creador de animaciones, un puntero, un contador, un indicador de porcentajes, un cuentakilómetros, un *slider*, etcétera.

DXVU METER 1.0.44

Control *ActiveX* que monitoriza los dispositivos de sonido. Controla cualquier dispositivo configurado como una fuente de grabación y muestra los niveles de sonido como un osciloscopio estándar o un medidor *VU*. El *VU* muestra el modo de trabajo calculando el valor *RMS* del sonido monitorizado, y el osciloscopio trabaja tomando muestras de los datos y mostrándolos en forma de ondas.

NCTWAVPLAYER 1.07

Control ActiveX para editar ficheros *WAV* y reproducir ficheros *MP3*.
Cuenta con controles de volumen y propiedades, y un editor visual que permite copiar y pegar partes de ficheros, realizar mezclas y añadir efectos. Carga rápidamente los ficheros de sonido y soporta los formatos *DSP, Microsoft GSM, MPEG-3, PCM*, etcétera.

OCTOPOD FOR C++ 3.0

Entorno de desarrollo integrado para crear aplicaciones en *C++* que permite crear aplicaciones pequeñas, compactas que no necesitan librerías adicionales. Su interfaz de usuario presenta características de *Corel Draw* y *Visual Basic*. Cuenta con un "explorador de proyectos" que analiza el código fuente y muestra todas las propiedades, métodos y funciones definidas en un programa.

TIFFDLL2 V3.17

Funciones *ActiveX* para añadir texto a ficheros de imágenes. Además de incorporar comentarios a cada imagen, permite superponer otras nuevas para crear imágenes superpuestas y marca de agua. En cuanto al texto, puede ser insertado en cualquier localización y en diferentes formatos.

HERRAMIENTAS Y UTILIDADES

CLONECD 2.1.1.1

Potente duplicador de *CD-ROM* que contando con el equipamiento adecuado, puede trabajar en modo *RAW* y duplicar discos protegidos, creando verdaderas copias idénticas de los *CD* (la mayoría de los modernos *CD-ROM ATAPI* y algunos *SCSI* pueden leer la información oculta).

DR. HARDWARE 99 V4.00E

Utilidad de diagnóstico y monitorización del equipo que realiza varias pruebas de rendimiento. Ofrece completa información sobre el del sistema, la gestión de la memoria, los ficheros, el procesador, la *BIOS*, el disco duro, los dispositivos *SCSI, PCI, GDI*, los *chipset* y las librerías *DirectX*.

FINEPRINT 2000 4.00 BUILD 15

Controlador de impresión que permite sacar múltiples páginas en una misma hoja.
Cada hoja puede contener 1, 2, 4 u 8 páginas de cualquier tipo de documento, manteniendo su estructura original. Ayuda a transportar

documentación, ahorrar espacio y disminuir el consumo de papel y tinta. Puede imprimir ambas caras, utilizar cualquier tamaño de papel, diferentes diseños y trabaja con todas las impresoras que soporten *Windows 95/98*.

ULTIMATE ZIP CRACKER 4.1

Recupera las contraseñas de ficheros *ZIP* y *ARJ*. Utiliza varios métodos, como *Brutal Force Attack*, *Dictionary Search* y *Birthday Search* para ayudar a recuperar claves de acceso perdidas. Permite especificar el rango de longitud de la contraseña, el tipo de caracteres que contiene y el método de búsqueda.

WINHEX 9.4

Completo editor hexadecimal que soporta los formatos *FAT16*, *FAT32* y *NTFS*.
Ofrece una gran rapidez de trabajo, necesita poca memoria y es muy fácil de usar. Puede importar todos los formatos del portapapeles, realizar búsquedas y sustituciones, generar números aleatoriamente, y borrar ficheros definitivamente.

WINZIP 8.0

Compresor y descompresor de archivos, rápido y fácil de usar.
Cuenta con soporte interno para los formatos más populares de *Internet*, como *TAR, ZIP, UUencode, XXencode, BinHex, MIME* y ficheros comprimidos *Unix*. También soporta los archivos *ARC, ARJ* y *LZH* pero utilizando programas externos. Esta versión permite instalar temas y salvapantallas, incluye mejoras en las funciones de ayuda, opciones de correo electrónico, menús contextuales y barras de herramientas similares a las del explorador de *Windows*.

IMPRESCINDIBLES

AntiViral Toolkit Pro 3.0.129
Copernic 2000 v4.5
Internet Explorer 5.01
Netscape Communicator 4.73
Panda Antivirus 6.17.00 Platinum
Paint Shop Pro with Animation Shop 6.02

REDES

ADVANCED ADMINISTRATIVE TOOLS 4.0

Monitor de seguridad de redes. Permite detectar problemas de seguridad en sistemas de *Internet*, *intranet* y *extranet*. Además, cuenta con recursos para analizar los problemas y solucionarlos. Incluye un verificador *proxy*, un verificador de listas de correo electrónico y utilidades que identifican los servicios activos y monitorizan *CGI´s*.

ALLEGROSURF NETWORK 3.0.1.0

Acelerador de Webs y servidor *proxy* de filtrado de contenidos. Permite que varios ordenadores conectados a una *LAN* puedan compartir una misma conexión a *Internet*. Soporta la mayoría de los protocolos de *Internet*, incluyendo *DNS, FTP, NNTP, POP3, SMTP, SOCKS4, SOCKS4a* y *SOCKS5*.

GATEKEEPER 3.5

Permite a los usuarios de una *LAN* acceder a *Internet* utilizando un mismo equipo. Este único equipo puede ser conectado a *Internet* a través de módem, una conexión *ISDN* o utilizando una segunda tarjeta de red *NIC* (*network interface card*).

ICB 2000 V4.0.5

Analiza y muestra información sobre distintos parámetros de conectividad de una red. Analiza la velocidad, el ancho de banda y otros aspectos, mostrando de forma gráfica resultados como la velocidad y el rendimiento medio de la conexión a *Internet* y del servidor. Ofrece numerosas funciones para hacer un seguimiento de la conectividad de red entre dos ordenadores conectados en una *LAN* o a través de *Internet*.

RELAYFAX 2.2.0.0

Cliente/servidor capaz de enviar mensajes de correo electrónico a un fax y viceversa. Se integra con el sistema de correo para proporcionar a los usuarios de una red funciones de fax desde sus ordenadores, utilizando la opción imprimir de cualquier aplicación *Windows*. Ofrece detección automática de módems, soporte para ficheros adjuntos y acceso *RAS*.

WORKGROUPMAIL 5.02

Servidor *POP3* y servicio *MAPI* que proporciona correo a través de *Web*.

DOCUMENTACIÓN/ TUTORIALES

IBUILDING A JAVA APPLET

Completo curso sobre la creación de *applets Java*. Guía paso a paso en el proceso de desarrollo *Java* u en los *applets*.

INTERCOM 2.1

Colección de aplicaciones que muestran los aspectos esenciales de los protocolos de las comunicaciones de *Internet*.

MICROSOFT FRONTPAGE 2000 TUTORIAL

Completo tutorial sobre *Microsoft FrontPage 2000*. Abarca aspectos como el diseño del sitio y la publicación de las páginas.

ATENCIÓN:
En caso de problemas con el CD-ROM envíelo por correo ordinario, a la atención del SERVICIO TÉCNICO DE SÓLO P., incluyendo en el interior del sobre sus datos personales, a la siguiente dirección:
C/ San Sotero, Nº 5, 1º Planta,
28037 (Madrid)

Windows 2000 Server contra Linux

Alfredo Vegas.
Administrador de Sistemas.

Microsoft quiere mantener la supremacía de los grandes sistemas operativos, desafiada en los últimos tiempos por _Linux_. Veamos las armas que usan ambos en esta guerra.

INTRODUCCIÓN

Microsoft ha presentado sus cuatro versiones de *Windows 2000: Professional*, *Server*, *Advanced Server* y *DataCenter Server*, orientada cada una a un segmento específico del mercado. En este artículo, se trabajará con la versión *Server*, que incluye todas las características principales de la familia, pues la *Professional* está mas orientada a estaciones de trabajo aisladas o clientes, y las versiones superiores tienen sus ampliaciones principalmente en el campo del rendimiento a gran escala.

En el otro lado del *ring*, colocaremos a *Linux Mandrake 7.0-2*, que se está convirtiendo en una importante opción para la mayoría de los que han perdido interés por *RedHat*, merced a su gran cantidad de paquetes y la buena integración entre los mismos. Esta versión funciona con el *kernel 2.2.14-15*, y como gestor de ventanas principal, utiliza el *KDE 1.1.2*. Pero aunque tomemos esta distribución como base, todo de lo que digamos de *Linux* podemos extenderlo a todas sus distribuciones.

ASPECTOS GENERALES

Parece que en *Microsoft* se han fijado bastante en las posibilidades de *Linux*, pues las herramientas de su nuevo sistema operativo recuerdan bastante a las de su competidor, como las cuotas de ocupación del disco para los usuarios, aunque sólo sea en *NTFS*.

Las principales armas que ha usado *Linux* para enfrentarse al dominante *Windows* han sido el precio y la estabilidad. Un precio prácticamente nulo, o irrisorio frente a los de la familia *Windows*, y sobre todo dentro de la gama *Windows 2000*. Por este lado, la batalla sigue ganada por *Linux*, y presumiblemente será así por mucho tiempo.

Windows se ha caracterizado desde su creación por la facilidad de uso. Aunque esta facilidad no permite entrar profundamente en el sistema operativo, sí es perfectamente válida para dar los primeros pasos, de manera que la progresión en la dificultad de aprendizaje es más paulatina.

Esta facilidad es en parte consecuencia, de la homogeneidad del producto. Casi el 90% de las aplicaciones existentes para la plataforma

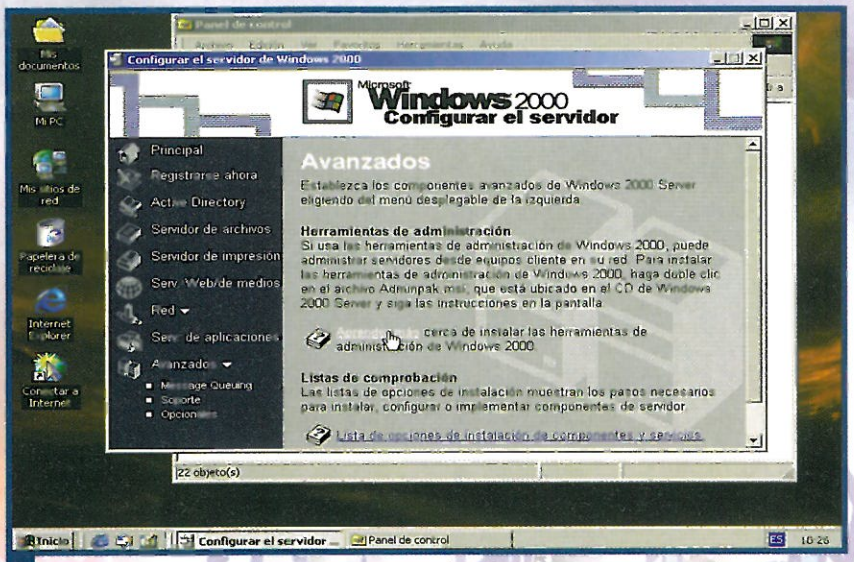


Figura 1.- Pantalla desde la que se procede a realizar la configuración gráfica de los diferentes servicios que proporciona el servidor.

tienen un sistema de uso similar, siendo esto propiciado por *Microsoft*, con la publicación de guías para programar la interfaz gráfica. Para aumentar todavía más esta facilidad de uso, se ha añadido una gran cantidad de asistentes, que si bien no están disponibles para ciertas características muy específicas, sí sirven para facilitar la puesta en marcha de los diferentes servicios, de manera que el tiempo transcurrido entre la primera instalación y su operatividad, es relativamente corto.

El otro aspecto en que sigue manteniendo la delantera *Windows 2000*, es el número de aplicaciones que corren en su sistema operativo. Esto es muy importante, pues no sería la primera vez que aparece un gran sistema operativo que, al no disponer de aplicaciones, se queda rápidamente en el olvido.

Obviamente, esto no va a ocurrir con *Linux*, pero el número de aplicaciones (sobre todo grandes aplicaciones, como *BBDD*, *suites* de oficina, retoque gráfico, etcétera) que corren en *Windows 2000* es superior al de *Linux*. La buena noticia, es que esta diferencia está menguando, gracias al comportamiento de compañías como *IBM* y *Corel*, que al presentar programas para *Linux*, arrastran a otras empresas a interesarse por esta plataforma.

Windows 2000, más fácil de usar y con más aplicaciones

En cuanto a la estabilidad, todavía no se puede hablar demasiado de *Windows 2000*, pues aún no se han hecho pruebas exhaustivas, como la de mantenerlo funcionando varios meses, sin bloqueos que hagan necesario reiniciar el equipo. La estabilidad de *Windows 2000* por períodos cortos de tiempo es bastante buena, sobre todo si lo comparamos con *Windows 9X*, aunque se nota que en este caso es un digno sucesor de *Windows NT*. Gran parte de esta estabilidad consiste en que, aunque se estropee algún programa, el sistema sigue en funcionamiento, necesitando tan sólo cerrar el programa en cuestión. La famosa pantalla azul casi ha desaparecido en este sistema. *Windows 2000* se acerca bastante a la estabilidad de *Linux*, aunque todavía tiene que demostrar que es capaz de estar un año funcionando sin caerse. *Windows* se ha acercado mucho a *Linux*, en la supresión de reinicios necesarios para configurar el sistema.

Un campo en que están bastante igualados, es en el del soporte de dispositivos... que en ningún caso es muy bueno. En *Linux*, nos encontramos con la importante traba de los fabricantes de dispositivos, que no liberan el acceso a los mismos, o no crean controladores para *Linux*. Un ejemplo bastante clásico de este caso, son los *Winmodem*, que hoy por hoy, y salvo raras excepciones, sólo funcionan con *Windows*.

Por el otro lado, *Microsoft* se ha acercado a *Linux* en su peor aspecto, pues ahora nos podemos encontrar con gran cantidad de dispositivos que tenían *drivers* para *Windows NT* o *9X*, y que ahora no son reconocidos por *Windows 2000*. Y es que *Microsoft* ha redefinido la forma de comunicar los controladores con el *kernel*, pasando a *WDM* (*Windows Driver Model*), un sistema teóricamente capaz de hacer compatibles los *drivers* entre *Windows 98* y *2000*, y que obliga a los fabricantes a reprogramar estos *drivers*, lo que no siempre hacen.

En cuanto a la ayuda al usuario, la victoria se la lleva con diferencia *Windows*. Las empresas que venden *Linux* están creciendo mucho, y saben que han de cuidar mucho esta asistencia. Pero la facilidad que tiene el mundo *Windows* para obtener asistencia es aún muy superior a la de *Linux*.

Aquí destacan los cursos de certificación que imparte *Microsoft* y sus aliados, a diferencia del mundo *Linux*, donde la enseñanza es escasa, sin un sistema de certificación centralizado. La mejor y principal manera de aprender en *Linux*, es leer

Figura 2.- Gestor de escritorios de *KDE* en *Linux*. Selección del tema del escritorio.

grandes cantidades de *howtos* (cómo se hace), estar continuamente conectado a los grupos de noticias, y rezar por que alguien con más experiencia o conocimientos que nosotros, haya tenido un problema similar al nuestro, y ponga una solución en las *news*.

La diferente capacidad de creación de *software* que puede tener un grupo heterogéneo de programadores colaborando entre sí como iguales, frente a un grupo de trabajo perfectamente estructurado, se traduce en la sensación de unidad y cohesión que proporciona *Windows 2000,* aún inasequible para Linux.

Y esta diferencia podría aumentar, debido a la gran proliferación de empresas que se basan en *Linux* para sacar sistemas operativos, y las extensiones añadidas por cada una de ellas al núcleo del sistema, lo que aumenta la heterogeneidad del producto. Por no hablar de los particulares a los que no gusta cómo se hace tal o cual cosa, y sacan su propio programa adaptado a sus preferencias.

Si esta diversidad tan creciente es beneficiosa o perjudicial para *Linux*, sólo el tiempo lo dirá. Pero por este camino, no pasará mucho tiempo hasta que dos expertos en *Linux*, cada uno trabajando en un sistema complejo, no puedan hablar entre sí, porque hacen uso de herramientas completamente diferentes. Aunque lo que se pierde en cohesión, se gana en libertad, y al final, puro *Linux*: todo a gusto del usuario.

INSTALACIÓN

El primer paso necesario para evaluar un sistema operativo, es su instalación. Un arduo sistema de instalación, puede llevar al desánimo al más preparado. Hasta hace unos meses, era el caso de *Linux*... y todavía lo sigue siendo en ciertas distribuciones. Pero ya han llegado a nuestras pantallas los instaladores gráficos, los sistemas de búsqueda inteligente de dispositivos, la reducción de preguntas, y la disminución del tiempo de instalación.

Ambos sistemas operativos ya son capaces de hacer la instalación desde el propio *CD,* sin necesitar discos de arranque. Esto es un avance sobre las instalaciones en las que había que crear los discos de arranque, aunque el resto de la instalación se realizase desde el *CD*.

La instalación gráfica en *Windows* hace ya mucho que está implementada, mientras que en *Linux* sólo ha comenzado a extenderse recientemente por las diferentes distribuciones. La instalación gráfica conlleva, sobre todo, una riqueza visual que no se tenía con las instalaciones basadas en modo texto.

Sólo el tiempo dirá si la heterogeneidad de Linux es perjudicial o provechosa

Pero más importante que la impresión visual de la instalación, es la capacidad del sistema para detectar todos los dispositivos, con la mínima intervención por parte del usuario. En la detección del *hardware*, *Windows* se adelanta algo, principalmente por la gran cantidad de controladores de que dispone, ya que casi todos los fabricantes, programan sus dispositivos para *Windows*.

Lo malo, es que no todo se detecta siempre. *Linux* también detecta la mayoría del hardware de que se dispone, pero aún es necesaria la intervención del administrador, sobre todo tras la instalación, donde la mayoría de las veces es necesario proceder a recompilar el *kernel,* para ajustar el rendimiento a nuestro sistema. Además, en *Windows 2000*, ya hay posibilidad de hacer uso de los dispositivos que emplean el puerto *USB* para comunicarse con el ordenador, posibilidad de la que no disponía *NT*. Hay que recordar que *Linux* no soporta *USB*, y tendremos que esperar hasta la versión *2.4* del *kernel* para poder emplear este tipo de dispositivos.

La velocidad de instalación, en cambio, es bastante superior en

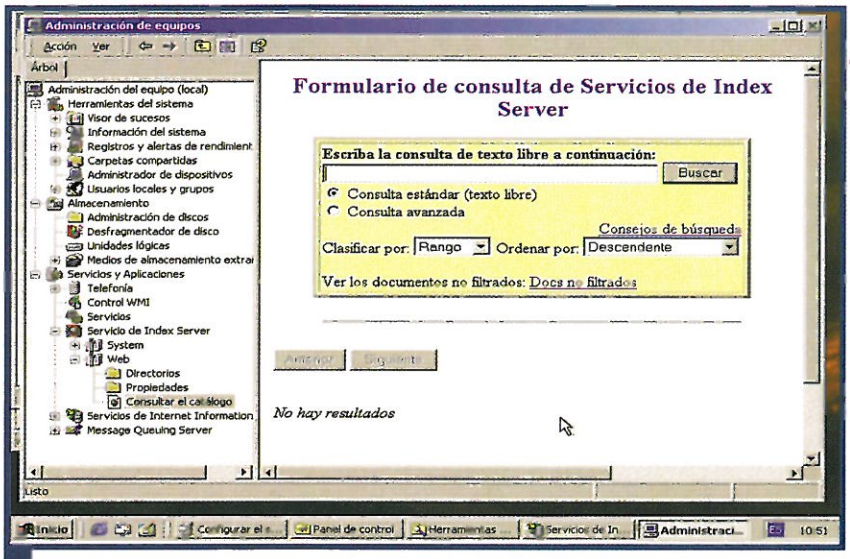


Figura 3.- Consola de administración de los componentes, en la que se puede ver el funcionamiento del buscador de ficheros.

Linux, pues tarda menos en detectar los dispositivos hardware y en copiar todos los paquetes de soporte lógico al disco duro. Incluso haciendo una instalación completa, de aproximadamente 1'5 *GB* de espacio en disco, el tiempo de instalación puede ser la mitad del necesario en *Windows*.

La instalación y actualización del *software* por red es una característica de ambos sistemas. Quizá se pueda considerar algo mas preparado para redes a *Windows* y sus servicios de instalación remota, que facilitan mucho la labor a los administradores de grandes redes, en las que pueden llegar a instalar más de 100 veces el sistema operativo. Debido a esta gran cantidad de instalaciones, es importante que una vez hecha la primera, las siguientes sean una copia de ésta.

En *Linux*, una de las forma más empleadas, es hacer una copia del disco de instalación en el disco duro de otro equipo. Por desgracia, este sistema es útil tan sólo para equipos que tengan exactamente las mismas características y dispositivos. La otra opción de instalación por red, es ir leyendo los paquetes en un servidor remoto, e ir instalándolos en el propio ordenador.

GUI Y ENTORNO

Actualmente, la comunicación hombre-ordenador se realiza a través de interfaces gráficas. Estas interfaces, se basan en el principio de ventanas que se sitúan en un escritorio, en las que se va mostrando la información al usuario.

Windows, como su propio nombre indica *-ventanas-*, se basa completamente en este sistema para interactuar con el usuario. En cambio, *Linux* aúna los dos sistemas,la interfaz de comandos en modo texto, y la interfaz gráfica. Aunque *Windows* también dispone de un sistema de control por comandos, y dado que en ambos sistemas operativos es bastante similar, nos vamos a centrar en comparar la interfaz gráfica.

Uno de los problemas -o ventajas- que tiene *Linux,* es la gran variedad de entornos de los que dispone el usuario para gestionar el escritorio. Esto puede suponer un complicado, porque con esta variedad, es difícil saber cuál es el gestor de ventanas más adecuado. Por esto, lo normal es que casi todos los usuario se acostumbran a usar el entorno que viene por defecto en la distribución elegida. La ventaja, es que siempre vamos a encontrar un entorno que se adapta a nuestras necesidades.

Linux es, posiblemente, la plataforma con más lenguajes de programación

Lo peor que tiene esta abundancia de gestores de ventanas, es la proliferación de aplicaciones preparadas u optimadas para habitar en tan sólo uno de ellos, funcionando de manera incorrecta, o sencillamente no haciéndolo, si intentamos hacer que corra en otro entorno.

Así encontramos los entornos *KDE, Window Maker* y *GNOME*, cada uno de ellos con una serie de herramientas específicas, que se pueden diferenciar por el comienzo del comando (*'k'* para *KDE*, 'wm' para *Window Maker*, y *'g'* para *GNOME*), que en un afán de tener la interfaz más avanzada, están perdiendo parte de la libertad de elección.

Muy atrás quedan ya los sencillos gestores como el *twm* o *fvwm95*, que lo único que pretendían era tratar de una forma coherente las ventanas que aparecían en el escritorio, permitiendo incluso cambiar el gestor de ventanas en mitad de la sesión. Algunos de estos gestores están logrando también que el sistema se pase más tiempo de proceso con el gestor de ventanas, que con los programas

que se ejecutan en su interior, debido a su complejidad.

A pesar de esa gran cantidad de entornos gráficos o gestores de ventanas existentes en *Linux*, aún sigue siendo un sistema más orientado a la línea de comandos que al aspecto gráfico. Esto se comprueba en que gran parte de las aplicaciones gráficas de relevancia, son *front-ends* gráficos a programas, que pueden funcionar perfectamente en modo de línea de comandos. Lo que se agrava, además, porque el sistema de ventanas de *Linux* es, al fin y al cabo, otro programa más que se ejecuta dentro del sistema, pero sin estar completamente integrado con el *kernel*, con lo que la perdida de rendimiento es notable.

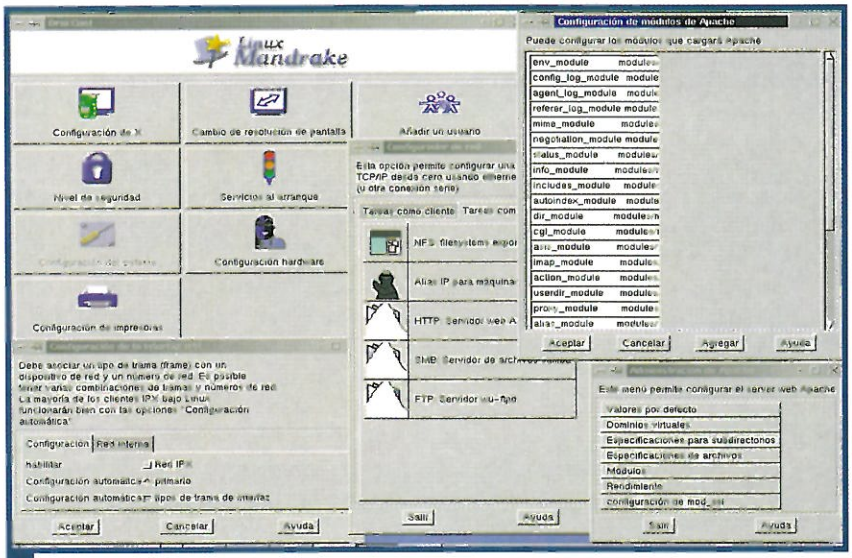


Figura 4.- Configuración del sistema y de los diferentes servidores, como *Apache*, desde *LinuxConf* y *DrakConf* (herramienta Mandrake de configuración).

La integración de Windows con su interfaz gráfica es total

En cambio, la integración de *Windows* con su interfaz gráfica es total, dejando la línea de comandos para cuestiones muy específicas. La "amabilidad" del entorno es superior en *Windows*, aunque la buena gestión que hace *Linux* con los escritorios virtuales, ya la quisiera *Windows* para sí.

El lenguaje del entorno, también es un punto importante. En *Windows 2000*, viene incorporado un sistema de gestión de idiomas (*MUI, Interfaz de Usuario Multilingüe*), lo que nos permite gestionar con varios idiomas. Por lo general, el idioma que más utilizaremos, el español, tiene una traducción muy buena. En *Windows*, todos los menús, asistentes, documentación, y el entorno en general, están traducidos. Debido al sistema de librerías de enlace dinámico (*DLL*) con el que trabaja *Windows*, también muchos programas que vienen de fuera pueden tener los menús y ciertos botones traducidos.

Todo esto viene también facilitado por el *NLS* (*Native Languaje Support*), un sistema dentro del propio sistema operativo, que engloba, además de las características del lenguaje, detalles como la ordenación de las fechas, las horas, días de la semana, moneda, etcétera, de manera que los programas le dicen al sistema que presente un dato como fecha, y él mismo elige la forma de hacerlo, dependiendo de la configuración del usuario.

Para facilitar esta internacionalización del *software*, *Windows 2000* trabaja internamente con *Unicode*, un nuevo estándar de presentación de caracteres. La principal diferencia con el conocido *ASCII*, o *Código Estándar Americano para el Intercambio de Información*, es que pasa de usar 8 *bits* a 16 *bits* para cada carácter. Así se logra que tengamos a nuestra disposición 65.000 caracteres diferentes, de los que se emplean en la actualidad unos 40.000, en su mayoría usados por las escrituras de símbolos asiáticos.

Tanto el sistema de archivos *NTFS* como los archivos de texto, ya se pueden escribir en este formato, pero con el lastre de que, como *Linux* aún no lo incluye, los archivos que guardemos en este sistema, van a ser difícilmente legibles por *Linux*.

En *Linux*, a pesar de la abundancia de proyectos para trasladar el sistema a nuestro idioma, entre los que cabe destacar el proyecto *LuCAS* (*Linux en CAStellano*) dentro de la *Asociación Española de Usuarios de Linux* (*HispaLinux*), todavía queda mucho que hacer. *LuCAS* está traduciendo las páginas de manuales y diversa documentación, y ayuda a las diferentes distribuciones a que éstas se realicen en castellano.

El sistema de gestión de comunicación entre tareas en la pantalla de trabajo, es por ahora, otra de las victorias de *Windows*. Sus sistemas *OLE* y *DDE* permiten que el intercambio de información entre las aplicaciones activas sea tan sencillo como "arrastrar y soltar" los datos entre una aplicación y otra. Y el uso del portapapeles también es mucho

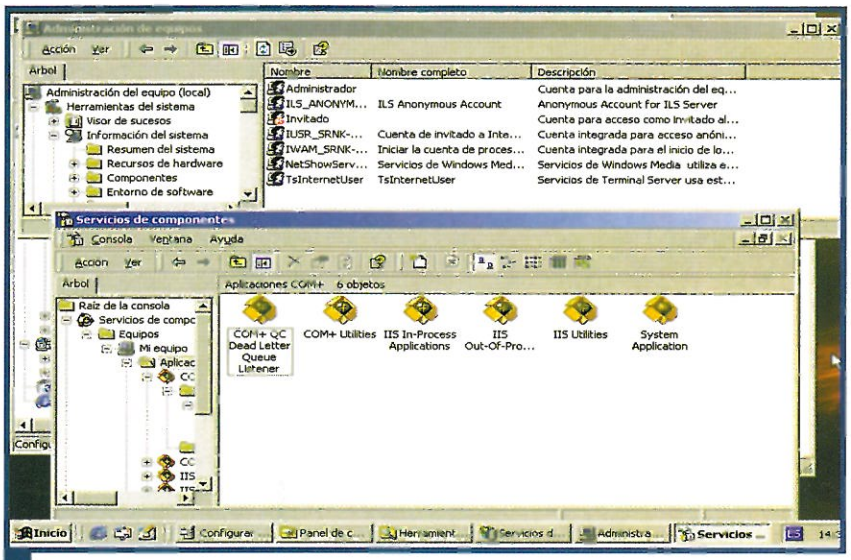


Figura 5.- Consolas para la administración de usuarios y grupos de usuarios, junto a la consola de administración de servicios *COM+*.

más sencillo y potente, siendo capaz de albergar casi cualquier tipo de datos.

La traducción al español de Windows es muy buena

En *Linux*, la información que fluye entre los diferentes componentes del escritorio, se limita casi exclusivamente a líneas de texto. Algunos de los modernos gestores de ventanas se están preparando para facilitar el intercambio de información al estilo de *Windows*, pero esto es muy complicado, ya que las aplicaciones no están preparadas para funcionar así.

Una de las grandes ventajas que tiene *Linux* sobre *Xwindow*, es la posible separación entre el ordenador que se encarga de gestionar el escritorio, y el que se encarga de ejecutar los programas. Esto es lo que se engloba dentro de la tecnología cliente-servidor de *Xwindow*. Esta separación, permite tener un ordenador muy potente haciendo las veces de servidor de procesador, y otros más pequeños, haciendo de servidores de pantalla.

En respuesta, *Windows 2000* incorpora un servicio de terminales remotos (*WTS*), con el que también se tiene un servidor central que realiza el proceso de los programas, y unos terminales ligeros que presentan la ejecución al usuario. Esto es un gran cambio respecto al anterior sistema de *Windows*, en el que solo un usuario podía emplear la máquina en modo gráfico. Visto de forma sencilla, cada uno de estos clientes ligeros, lo único que usan como suyo es el teclado, el ratón y la pantalla.

HERRAMIENTAS INCLUIDAS

Aquí, la victoria de *Linux* es aplastante, ya que el número de herramientas, programas de apoyo y utilidades que incorpora, (aunque varíe entre instalaciones), es enorme. Sería muy discutible definir qué pertenece al sistema operativo, y qué "al resto" de programas. Aquí es donde toma sentido la gratuidad de *Linux*, pues un sistema operativo por sí solo, es poco útil.

El problema de *Windows* no es la falta de aplicaciones -más bien al contrario-, sino el precio de las mismas. En *Linux*, tan sólo hay una o dos aplicaciones dentro de cada campo, pero éstas son gratuitas. También es cierto que dentro del entorno *Windows* se encuentran aplicaciones con mayor calidad, de aspecto más profesional y mejor acabado. Por suerte, esta diferencia se está eliminando, pues cada día son más las empresas que sacan sus productos también para *Linux*, aunque no con la gratuidad a la que estaban acostumbrados los usuarios de *Linux*.

Las aplicaciones de *Windows 2000* son muy limitadas, a excepción de los servidores, que son algo más completos. Hay dos razones principales para explicar esta escasez. La primera es, que como prácticamente todo en esta plataforma es de pago, incluir más aplicaciones podría elevar demasiado el precio, con la circunstancia de que muchas de las aplicaciones no serán útiles para muchos usuarios. La otra, proviene de las quejas formuladas por varios creadores de *software*, referidas al hecho de que al incluir una herramienta similar a su producto con el propio sistema operativo, sus ventas se podrían ver muy afectadas. Posiblemente, el mayor ejemplo de hasta dónde se puede llevar este extremo, es la inacabada pugna entre *Microsoft* y *Netscape*.

Este problema no se da en *Linux*, al menos de momento, pero empiezan a verse los primeros atisbos en la queja que han manifestado los creadores del servidor *WEB* de *Apache* al propio *Linus Torvalds*, por la pretensión de éste de incluir un demonio servidor de *Web* en la próxima versión *2.4* del *kernel*, que se espera para finales del verano.

LOS JUEGOS

Una de las lacras de la familia de *Windows NT*, es que siempre se ha considerado que estos sistemas eran "serios", sin posibilidad de ejecutar juegos en los mismos. Se dijo que *Windows 2000* iba a incluir la última versión de *DirectX* para *Windows*, la *7.0*. Pues es cierto, está incluida y soportada por el sistema, pero eso no quiere decir que funcionen los juegos. Por desgracia, los que hemos intentado probar, y que hacían uso de esta versión de *DirectX* o anteriores, y que sí han funcionado con *NT 4.0* (usando un parche no oficial de *DirectX 7* para *NT*), no lo han hecho con *Windows 2000*, generando errores de protección. El juego más empleado en *Windows 2000* seguirá siendo el solitario.

Linux supera a Windows en herramientas y utilidades

En cambio, *Linux* tiene más juegos, sobre todo desde que *Loki Games* realiza conversiones de varios juegos actuales de *Windows* a *Linux*. Los juegos que suelen venir con las distribuciones no tienen una gran calidad gráfica, pero sí mucho entretenimiento.

CONFIGURACIÓN Y MANTENIMIENTO

Aquí es donde más ha progresado *Linux*, aunque todavía tiene mucho que avanzar. Siempre se ha tenido a *Linux* como un sistema para informáticos, donde el resto de los mortales no tenían mucho que hacer. Esto se debe principalmente a la dificultad de configuración inicial y mantenimiento en *Linux*, y a la cantidad de problemas que se puede tener para que funcionen todos los dispositivos (las *news* están llenas de mensajes del tipo: *"No me funciona .....¿qué puedo hacer?"*.

Con la creación de paquetes como *"LinuxConf"* y *"WebMin"*, las labores de configuración se han simplificado mucho, ya que estas herramientas permiten editar los ficheros de configuración desde un entorno de menús que facilita sobremanera este mal trago. Aún así, siempre hay que acabar usando algún editor de texto para afinar nuestro sistema.

En cambio, la configuración de la mayoría de los aspectos del sistema en *Windows 2000* es mucho más fácil, gracias a los asistentes, de los que hay uno para casi cada parte del sistema que se pueda configurar, con lo que respondiendo a unas preguntas, se puede tener rápidamente funcionando un servidor *Web* o un *FTP*. Y claro, siempre se puede entrar directamente en las herramientas de configuración para variar algunos de los detalles que estos asistentes puedan tomar por defecto. Estas funciones de configuración se centralizan en *MMC (Microsoft Management Console)*. También hay una gran cantidad de asistentes de solución de problemas, que ahorra llamadas al servicio técnico, y el consiguiente gasto de tiempo y dinero.

Típico de *Windows* era la "suciedad" que podía llegar a acumular el sistema, es decir, la gran cantidad de librerías y restos de programas acumulados tras instalaciones y desinstalaciones, que variaban las librerías del sistema por otras propias, con la consiguiente pérdida de estabilidad. Este problema parece haber sido solucionado con un nuevo sistema de seguimiento de instalaciones, así como con la comprobación automática de librerías de sistema, que si detecta algún cambio, lo deshace automáticamente.

Linux deja para el usuario todas estas labores de administración y mantenimiento, aunque tiene tantos problemas de sustitución de librerías, pues no es un sistema tan centralizado como *Windows*.

Resumen comparativo

Aspecto	Vencedor
Instalación	Linux
Soporte del hardware	Empate
Facilidad y comodidad	Windows 2000
Configuración	Windows 2000
Mantenimiento	Windows 2000
Programas incluidos	Linux
Programas disponibles	Windows 2000
Precio	Linux
Soporte al usuario	Windows 2000
Estabilidad	Linux
Seguridad	Windows 2000
Juegos	Linux
Programación	Linux
Redes	Empate
Internet	Windows 2000

PROGRAMACIÓN

Linux siempre ha sido una plataforma de trabajo y de desarrollo. Hoy ha crecido tanto, en ambos lados de la línea, que los usuarios se ven algo obligados a elegir entre dar a su sistema una orientación u otra. *Linux* siempre ha sido el gran amigo del lenguaje *C*, y también de

su primo, el *C++*. No obstante, siempre se ha abierto a todo tipo de lenguajes nuevos, lo que le convierte posiblemente en la plataforma con mayor cantidad de lenguajes de programación.

Windows 2000 tiene el problema de que no tiene ningún compilador de algún lenguaje integrado para programar. En la versión *Windows 2000* sí se han integrado algunos lenguajes interpretados, como *Perl, JavaScript, VBScript* y otros, a través del *WSH* (*Windows Scripting Host*), un sistema que integra los lenguajes interpretados dentro de un mismo entorno. Este sistema está mas orientado a la creación de *scripts* que sirvan de ayuda a la administración que a la creación de programas complejos.

En cambio, *Linux* dispone tanto de intérpretes, como de compiladores, y de sistemas integrados de desarrollo con los que crear cualquier tipo de aplicación. Y la documentación que viene con *Linux* es exhaustiva.

LA SEGURIDAD

Por lo general, *Linux* es un sistema operativo seguro, pero más que por la seguridad propia, un tanto limitada (los fuentes de casi todos los programas están disponibles para todo el mundo), es por la velocidad con la que salen parches para los agujeros que se vayan encontrando en la seguridad.

La seguridad de *Windows* consiste más en el desconocimiento de los *hackers* sobre las interioridades del sistema, pues por ningún sitio aparecen los fuentes. El problema, es que *Microsoft* suele ser mas lento publicando parches para los problemas detectados. Desde el principio

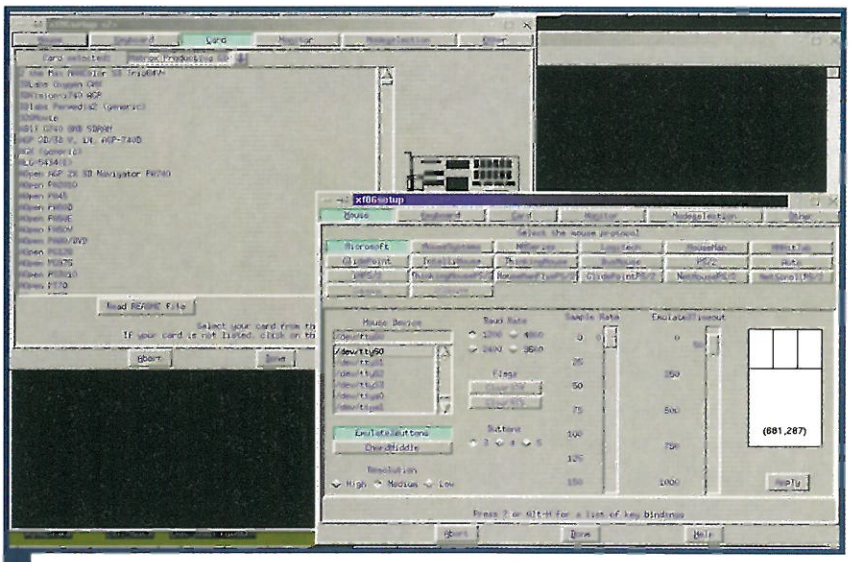


Figura 6.- Configuración del servidor de *XWindow* en modo gráfico. También se puede hacer desde una consola de texto.

del proyecto, *Microsoft* se quiso quitar la lacra de "sistema operativo poco seguro" achacada a sus productos. Por eso, ha trabajado mucho para aumentar la seguridad, no sólo para limitar visitas no deseadas, sino también con la seguridad de entrada en el sistema como usuario.

Windows es más seguro, y su gestión más rápida

Una de las características incorporadas a *Windows 2000* para aumentar la seguridad contra la entrada de intrusos, es el funcionamiento de las tarjetas inteligentes, similares a las de crédito en tamaño, y con un pequeño microprocesador que sirve para que el usuario se identifique frente al sistema sin necesidad de una contraseña. Esto aumenta la seguridad, pues ya no hay que teclear una clave que puede ser vista, sino que en todo momento se lleva encima la "llave" de nuestra cuenta. Al introducir la tarjeta en el lector, el sistema comprueba que quien la introduce es el dueño de la misma a través de la petición de un número *PIN*, y a partir de ahí comienza a comunicarse con la tarjeta por medio del protocolo de autenticación *Kerberos*, para terminar dando acceso al sistema al usuario autenticado. *Linux* tiene en la actualidad un proyecto para incorporar también este tipo de autenticación, pero de momento, *Windows 2000* es el único que lo implementa.

A la seguridad contra fisgones, se añade el *EFS* (Encrypt File System), una ampliación del *NTFS*, que hace que una partición de un disco pueda estar cifrada por un sistema de dos claves pública/privada, de manera que si se diera la circunstancia de que nos fuera sustraído el disco duro, no se podría acceder a la información en él contenida. Un servicio también añadido en esta versión, es el de un servidor de certificados con el propio sistema operativo. Esto es una facilidad importante para mantener el control de la seguridad dentro de la propia empresa. De momento, este tipo de servicio no esta disponible para *Linux*.

Por defecto, *Windows 2000* utiliza *Kerberos 5* para la autenticación de los servidores dentro de una red.

Este sistema se basa en billetes. Así, cada vez que un ordenador quiere salir a la red para emplear alguno de sus recursos, solicita un billete a un centro de distribución de claves, y usa el resultado para identificarse ante el resto de los servidores.

Para la seguridad en las redes, ambos sistemas operativos pueden usar la *VPN* (*Virtual Private Network*), que es una forma de transmitir datos de manera segura a través de un medio no seguro. Aunque *Windows 2000* puede usar varios protocolos para esto (*L2TP, PPTP*), el sistema más seguro y moderno para realizar estas redes privadas virtuales, es el *IPSec*, un conjunto de protocolos abiertos que han probado gran efectividad.

REDES Y SERVIDORES

Ambos sistemas operativos están muy orientados a trabajar en redes, sobre todo en *TCP/IP*, aunque soporten otros protocolos, como *NetBUI* o *IPX*. Y también ambos soportan *IP v.4* e *IPng* ó *IP v.6*. La versión *6* de *IP* proporciona un gran avance respecto a la versión *4*, especialmente por el aumento de posibles números *IP* y la gestión de la *QoS* (*Quality of Service*), por la que se puede decidir qué paquetes tienen prioridad para atravesar la red.

La integración de *Linux* con la pila de protocolos *TCP/IP* es muy buena, pues no en vano ésta se programó en *UNIX*, de donde desciende *Linux*. Esta pila define perfecta e independientemente de la máquina, la forma de trabajo en red, lo que hace que varios tipos de sistemas operativos puedan comunicarse con una sola red. *Windows*, en un principio, ha renunciado al protocolo de comunicaciones propio, y ha preferido adoptar *TCP/IP* como principal sistema de comunicaciones.

Ambos equipos pueden actuar tanto de *host*, como de servidor, e incluso de *router*. En este campo, *Windows* avanza bastante, y por fin se le puede cambiar la dirección *IP* sin tener que reiniciar, algo que *Linux* hacía desde hace mucho tiempo. Los servidores que incorporan ambos sistemas operativos son similares, como *HTTP, FTP, Telnet, Acceso en modo terminal gráfica, pop3, SMTP, DNS*, etcétera. En este aspecto están muy igualados, ya que ambos sistemas están muy orientados a actuar como servidores de datos dentro de una red.

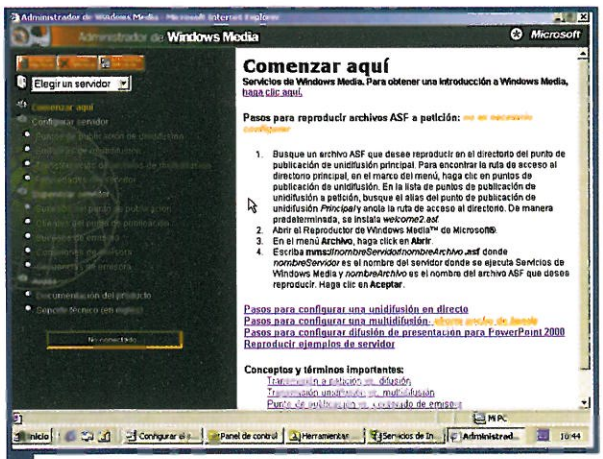


Figura 7.- Administración de los servicios *Windows Media*.

CONCLUSIÓN

Se comprueba que los dos sistemas se observan mutuamente, para incluir cada uno lo mejor del rival. *Windows* intenta tomar de *Linux* la estabilidad y la versatilidad, y *Linux* intenta tomar de *Windows* el aspecto y la facilidad de uso y mantenimiento. Decidirse es complicado, pues todo depende del uso que se dé al sistema. En la decisión, también cuenta mucho el precio, especialmente cuando se dispone de bastantes equipos.

Para un solo puesto, sin grandes necesidades de red, *Windows 2000* puede ser una buena elección, especialmente la versión *Professional*, pues tiene una gran cantidad de programas con una excelente calidad y facilidad de uso.

Para un entorno multiordenador, intercomunicado por red, la elección dependerá del tiempo que pueda dedicarse a la instalación, configuración y mantenimiento. Si se dispone de poco, lo mejor quizá sea *Windows 2000 Server*. Obtendrá más velocidad, aunque a cambio de perder control, pues es algo más críptico que *Linux*. Aunque se necesite más tiempo, con éste, el control ejercido sobre el sistema es mucho mayor, pues aunque requiera más conocimientos del administrador, le permite introducirse en todos los recovecos del sistema.

Por todo lo visto hasta ahora, no se puede dar un claro vencedor a uno de los dos, pues están muy igualados. Cada uno tiene características que superan al otro, pero el otro le contrarresta con otras. Por ahora, lo dejamos en empate, aunque esto pueda cambiar pronto, debido a la velocidad con que *Linux* se adapta y cambia. Quizá *Linux* sea más adecuado para sistemas en los que lo importante es el coste, la estabilidad y la fiabilidad, y *Windows 2000 Server,* para otros entornos donde la velocidad de trabajo sea muy importante. También es un motivo de peso los programas que vamos a usar en la red, y el sistema para el que estén preparados.

HASTA UN 45% DE DESCUENTO

PREPÁRATE PARA SER EL MEJOR WEBMASTER

- ✓ En España se ha duplicado el número de usuarios en los últimos 8 meses.
  Fuente: AIMC
- ✓ El 80% de internautas existentes en todo el mundo entran en la Red para realizar negocios.
  Fuente: FORRESTER RESEARCH
- ✓ 100 millones de personas navegan hoy.
  Fuente: Varios compilado por NUA INTERNET SURVEYS
- ✓ En 5 años el comercio electrónico supondrá 4,5 billones de pesetas tan sólo en los Estados Unidos.
  Fuente: IBID
- ✓ En el año 2.000 la gran mayoría de las transaciones no realizadas en dinero metálico serán mediante pagos electrónicos.
  Fuente: KILLEN ASSOCIATES

BOLETÍN DE SUSCRIPCIÓN

OFERTA VÁLIDA SOLO PARA ESPAÑA

SI, deseo suscribirme al **CURSO WEBMASTER DE PROGRAMACIÓN INTERNET**

CONDICIONES DE PAGO (MARQUE LA OPCIÓN DESEADA)

❑ 49 fascículos a **750 Ptas.**: **TOTAL 36.750 PTAS**.
❑ 4 carpetas para archivar fascículos a **950 Ptas.**: **TOTAL 3.800 PTAS.**
❑ 1 carpeta para archivar disquetes y CD´s a **1.250 Ptas.**
❑ Se regalará la versión de **FrontPage 2000** en el primer envío valorada en **29.750 Ptas.**

PRECIO TOTAL DE LA COLECCIÓN: 71.500 PTAS.

PAGO AL CONTADO:

❑ Ahorro del **45%. TOTAL PAGO AL CONTADO: 39.325 Ptas.**

PAGOS APLAZADOS:

❑ Pago en 3 entregas ahorro del **38%**: Cada entrega **14.780 Ptas.**
❑ Pago en 6 entregas ahorro del **30%**: Cada entrega **8.340 Ptas.**
❑ Pago en 12 entregas ahorro del **20%**: Cada entrega **4.770 Ptas.**

ENTREGA DEL FRONTPAGE 2000 EN LA ÚLTIMA ENTREGA

FORMAS DE PAGO:

❑ Con **cargo a mi tarjeta VISA** Nº ____/____ /____ /____ Caduca .....

❑ **Domiciliación bancaria**

Ruego a Uds. que se sirva cargar en mi ❑ cuenta corriente ❑ libreta de ahorro el/los recibos que le serán presentados por REVISTAS PROFESIONALES, S.L. como pago de mi suscripción al **CURSO WEBMASTER DE PROGRAMACIÓN INTERNET**

CÓDIGO CUENTA CORRIENTE

ENTIDAD	OFICINA	DC	Nº CUENTA

❑ **Cheque** a nombre de REVISTAS PROFESIONALES, SL.

❑ **Contra reembolso** del importe más gastos de envío.

❑ **Giro Postal** (adjunto fotocopia del resguardo)*.

*Esta forma de pago es exclusivamente para pagos al contado

Firma

ITILICE MAYÚSCULAS PARA RELLENAR EL PEDIDO

lombre y apellidos .....F. nacimiento.....
)omicilio ..... C.P. .....
:iudad ..... Provincia .....Telf. .....
-mail ..... Profesión .....

Curso de Programación con WAP (III)

Constantino Sánchez Ballesteros.
Técnico Superior Desarrollo Aplicaciones Informáticas.

En este artículo abordaremos una de las características que mayor potencia ofrece a *WAP*. Se trata de las variables, que dotan de dinamismo a las páginas *WML* desarrolladas para los dispositivos con soporte *WAP*.

INTRODUCCIÓN

Seguimos en este número con la serie dedicada a la tecnología *WAP*, la que nos permitirá ver contenidos publicados en *Internet*.

Ahora vamos a ocuparnos de ampliar las prestaciones de los móviles *WAP*. A pesar de las limitaciones actuales de la tecnología *WAP*, que dista todavía de acercarse a las posibilidades de otras formas de navegación en la Red más convencionales, podemos exprimir al máximo la capacidad de nuestro teléfono para implementar páginas interactivas de cierta complejidad. Para ello, utilizaremos las variables. En esta entrega, aprenderemos a definir y nombrar variables, su validación, asignación de valores, y sustitución.

Además, al final del artículo, vamos a estudiar la opción de crear, importar y editar nuestras imágenes compatibles con *WML*, desde el mismo entorno de desarrollo, sin tener que recurrir a un programa externo a *Nokia WAP Toolkit*.

VARIABLES

Las variables se pueden asignar a cualquier contenido *WML*, alcanzando de esta forma la adecuada flexibilidad en la creación de los *Decks* y *Cards* destinados a la visualización de contenido dinámico basado en las entradas que realice el usuario.

Otra característica de las variables *WML*, es que pueden ser utilizadas en lugar de las cadenas de caracteres *(strings)* y se sustituyen en tiempo de ejecución con su valor actual.

Hay que recordar, que una variable sólo es válida si contiene un valor no igual a una cadena vacía (nula), teniendo en cuenta que un valor no será establecido en una variable si éste es nulo o desconocido para el contexto del navegador.

DEFINIENDO VARIABLES: SETVAR

Cuando utilizamos *setvar* especificamos la variable que se establecerá

Figura 1. - Modificación del contenido de una variable.

en el contexto del navegador. Sin embargo, *element* será ignorado si el atributo nombre no se corresponde con un nombre de variable legal en tiempo de ejecución. En la Tabla 1 podemos ver con detalle los atributos para las variables.

En el siguiente ejemplo crearemos una variable llamada articulo con el valor por defecto ordenador. Además, se incluye la opción *Clear*, y al seleccionarla, se encarga de borrar el contenido de nuestra variable.

```
<card id="Producto" title="Selec-
    cione Producto">
<p>
Nombre del articulo:
<input title="Articulos"
    name="articulo" value="orde-
    nador"/>
<do type="accept"
label="Clear">
<refresh>
<setvar name="articu-
lo" value=""/>
</refresh>
</do>
</p>
</card>
```

En la Figura 1 podemos ver el resultado de este programa dentro del entorno de desarrollo.

NOMBRAR VARIABLES

Los nombres de variables bajo *WML* consisten en caracteres *US-ASCII* o guión bajo (_) seguido de más letras, dígitos o guiones (_). Cualquier otro carácter no es válido, y generará un error.

Las variables son sensibles a las mayúsculas y minúsculas. Por ello, *A* y *a* no son la misma variable, aunque se llamen del mismo modo. Los paréntesis son requeridos en cualquier lugar donde el final de una variable pueda inducir a error en el contexto de lo que se está escribiendo. Por ejemplo, cuando la variable acaba con un carácter ilegal o un espacio en blanco.

Los siguientes ejemplos muestran definiciones de variables legales:

- *$var.*
- *$(var).*
- *$(var:e).*
- *$(var:escape).*
- *$(var:unesc).*
- *$(var:N).*
- *$_X.*
- *$X32.*
- *$Test_9A.*

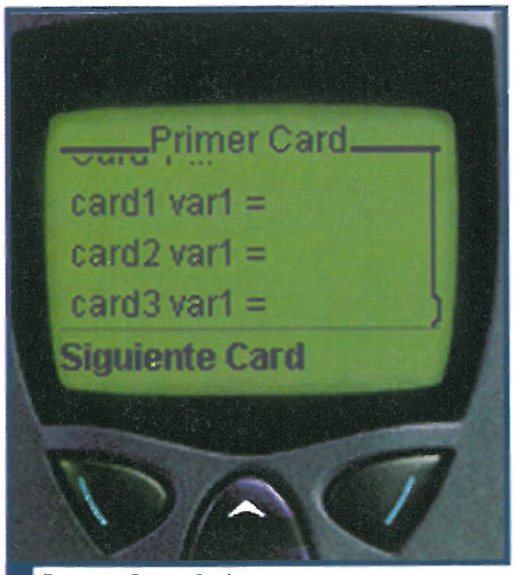


Figura 2. - Primer *Card*.

Para definir y nombrar variables, debemos seguir una sintaxis determinada

En casos especiales, como incluir un signo de dólar *($)* en un texto, debemos ponerlo como doble *($$)*. Dos signos *$$* son reemplazados por un simple *$*. Veamos un ejemplo:

```
Esto es un carácter $$.
```

Debería visualizarse como:

```
Esto es un carácter $.
```

Para incluir el signo *$* en una *URL*, debemos especificarlo mediante la secuencia de escape *%24*.

VALIDACIÓN DE VARIABLES

Dentro de un documento *WML*, cualquier cadena seguida de un signo dólar debe ser tratada como una referencia a una variable. Cada referencia debe utilizar la sintaxis del propio nombre de la variable.

Cada referencia debe colocarse en el texto de un *Card* o mediante los valores de atributos *%vdata* y *%ref*. Podemos obtener un *Deck* erróneo, si cualquier referencia a

TABLA 1. Atributos para las variables.

Atributo	Descripción
name=vdata	Nombre de la variable. Requerido.
value=vdata	Valor asignado a la variable. Requerido.

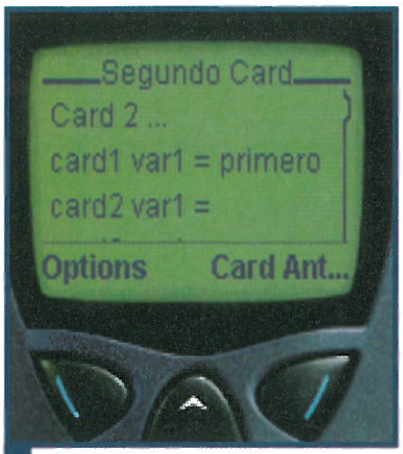


Figura 3. - Segundo *Card*.

una variable utiliza sintaxis inválida, o se coloca en una posición no válida. En el siguiente ejemplo se muestran utilizaciones no válidas de variables:

```
<!-sintaxis errónea ->
  El producto es $10.00
<!-posición no válida (en el
    atributo type) ->
<do type="x-$(type)"
    label="$type">
```

ASIGNAR VALORES

Hay varias formas de asignar el valor a una variable. Cuando una variable se define en el contexto del navegador su valor actual se actualiza automáticamente.

El elemento *setvar* permite establecer la variable fuera de ese contexto. *Setvar* puede especificarse dentro de los siguientes elementos *go, prev,* y *refresh*

Las variables también pueden asignarse en las siguientes situaciones:

- El elemento *input* asigna la variable identificada por el atributo name a cualquier información introducida por el usuario.
- El elemento *select* asigna el valor actual en el atributo value del elemento option elegido.

El elemento *setvar* especifica un nombre de variable y un valor:

```
<setvar name="location"
    value="$(X)"/>
```

El nombre de variable especificado en el atributo *name* se establece fuera del contexto de navegación.

CONVENCIONES

A continuación, se presentan varios puntos para tener en cuenta siempre que se trabaje con variables:

- Podemos cambiar valores de variables asignados con *WML* utilizando *WMLScript* y viceversa. Esto nos lleva a la conclusión de que *WML* y *WMLS cript* utilizan las mismas variables.
- Podemos asignar y editar variables en la vista del *WAP Toolkit,* como veremos más adelante.

Newcontext elimina los valores contenidos en las variables

- Podemos utilizar el atributo *newcontext* para eliminar todos los valores de variables del contexto actual del navegador.
- Si el atributo *href* del elemento *go* tiene referencias a variables, estas referencias sólo son resueltas dentro del contexto del navegador. Estas referencias no serán, en ningún caso, resueltas por elementos *setvar* dentro del elemento *go*.
- Si un elemento *go* contiene elementos *setvar* y los atributos *name* o *value* de *setvar* contienen referencias a variables, se resolverán dentro del contexto del navegador.

SUSTITUCIÓN DE VARIABLES

Podemos sustituir valores de variables en texto formateado *(PCDATA),* valores de opción *(vdata)* y atributos *href* en elementos *WML.* Hay que tener en cuenta que sólo se puede sustituir información textual, es decir, no es posible sustituir elementos o atributos. La sustitución de valores de variables se produce en tiempo de ejecución en el teléfono móvil del usuario.

Dado que la sustitución es simplemente una sustitución de cadenas, no afectará al valor actual de la variable. Si se hace referencia a una variable no definida, el resultado obtenido será una cadena vacía.

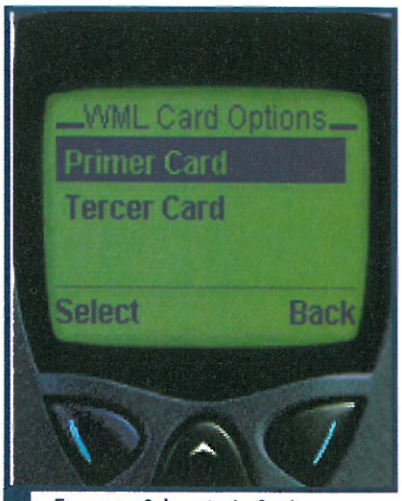


Figura 4. - Selección de *Card*.

LISTADO 1. Trabajando con variables

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
   "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<card id="card1" title="Primer Card" newcontext="true">
<p>
card 1 ... <br/>
<!— Las siguientes variables no se definirán hasta que
entremos en los otros Cards del Deck.
—>
card1 var1 = $(card1_var1) <br/>
card2 var1 = $(card2_var1) <br/>
card3 var1 = $(card3_var1) <br/>
<do type="accept" label="Siguiente Card">
<go href="#card2">
<setvar name="card1_var1" value="primero"/>
</go>
</do>
</p>
</card>
<card id="card2" title="Segundo Card">
<p>
card 2 ... <br/>
card1 var1 = $(card1_var1) <br/>
card2 var1 = $(card2_var1) <br/>
card3 var1 = $(card3_var1) <br/>
<do type="accept" label="Primer Card">
<go href="#card1"/>
</do>
<do type="accept" label="Tercer Card">
<go href="#card3">
<setvar name="card2_var1" value="segundo"/>
</go>
</do>
<do type="prev" label="Card Anterior">
<prev/>
</do>
</p>
</card>
<card id="card3" title="Card Anterior">
<onevent type="onenterforward">
<refresh>
<setvar name="card3_var1" value="tercero"/>
</refresh>
</onevent>
<p>
card 3 ... <br/>
card1 var1 = $(card1_var1) <br/>
card2 var1 = $(card2_var1) <br/>
card3 var1 = $(card3_var1) <br/>
<do type="prev" label="Anterior">
<prev/>
</do>
</p>
</card>
</wml>
```

El valor de las variables puede ser convertido de formas diferentes cuando se sustituyen. Una conversión puede ser especificada en una referencia de variable seguida por dos puntos (:). A continuación, se muestran varias formas de realizar conversiones:

- *$setvar* o *$(setvar):* El agente de usuario desestima la variable, utilizando el elemento *URL* dentro del contexto apropiado.

- *$(setvar:e)* o *$(setvar:escape):* Sustituye el valor de *setvar,* desestimando los caracteres no alfanuméricos de acuerdo con las convenciones del codificado *URL.*

- *$(setvar:N)* o *$(setvar:noesc):* Sustituye el valor de *setvar,* sin desestimar los caracteres no alfanuméricos.

El uso de una conversión durante la sustitución de variables no afecta al valor actual de la variable.

En el siguiente ejemplo, se muestra cómo mandar por correo un par de valores de tipo *name* al servidor *WEB* para que puedan enviarse los datos del cliente:

```
<go method="post"
     href="http://hostname/serv-
     let/dealer">
<postfield name="crear"
     value="levis"/>
<postfield name="pantalon"
     value="501"/>
</go>
```

En el Listado 1 podemos ver cómo referenciar variables, crearlas, manejar sus valores, etcétera. Se trata de un ejemplo completo con los detalles comentados en este artículo. En el ejemplo se utilizan los siguientes elementos *WML:*

- *card*
- *do*
- *go*
- *prev*
- *setvar*
- *onevent*
- *refresh*

El funcionamiento del programa es simple, pero funcional. Se asignan variables para cada *Card,* que en total son tres. Cuando se inicia el programa, las variables no contendrán ningún valor. En el momento de saltar de un *Card* a otro se asignarán valores a las variables del *Card* del cual acabamos de realizar el salto.

Cuando se ejecuta el programa podemos ver que las tres variables no contienen ningún valor. En este momento nos encontramos en el primer *Card.*

Toolkit permite examinar en tiempo de ejecución el contenido de variables

Las variables no tienen ningún valor en principio, ya que utilizamos el elemento *newcontext* para vaciar su contenido.

Figura 5. - Tercer card.

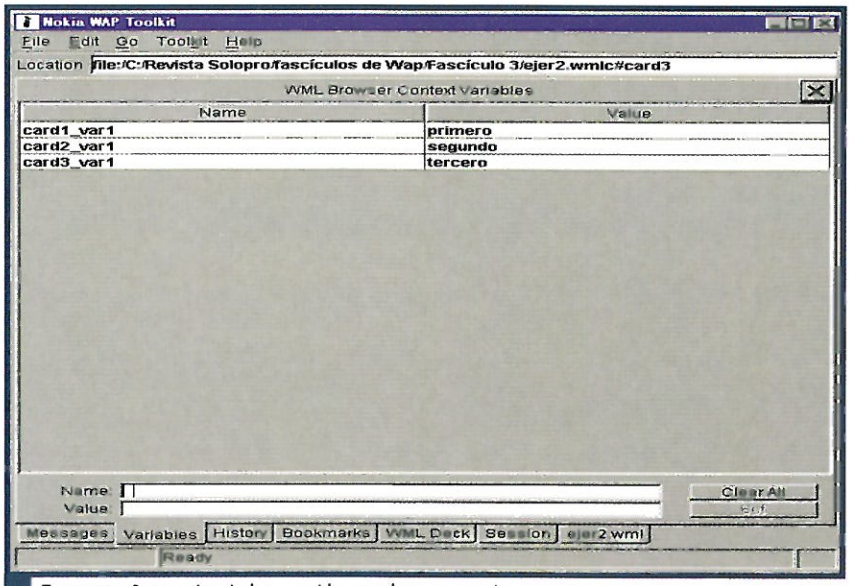


Figura 6. - Contenidos de las variables en el tercer *card.*

```
<card id="card1" title="Primer
    Card" newcontext="true">
```

Para visualizar el contenido de las variables, utilizamos la sintaxis comentada en apartados anteriores:

```
card1 var1 = $(card1_var1) <br/>
card2 var1 = $(card2_var1) <br/>
card3 var1 = $(card3_var1) <br/>
```

Si pulsamos sobre *Siguiente Card* vemos que la variable del primer *Card* ha pasado a contener el valor primero.

Este valor se ha asignado mediante la siguiente instrucción:

```
<setvar name="card1_var1"
    value="primero"/>
```

Pulsando sobre el botón *Options* aparece una pantalla en la que podemos seleccionar el *Card* al cual queremos dirigirnos. En este punto, los valores asignados a las variables permanecerán inalterables.

Si seleccionamos el tercer *Card* vemos que todas las variables contienen ahora un valor.

La instrucción que asigna valor a la tercera variable es la siguiente:

```
<setvar name="card3_var1"
    value="tercero"/>
```

VER EL CONTENIDO DE VARIABLES

En la pestaña variables del entorno de desarrollo de *Nokia* podemos visualizar qué valores contienen las variables definidas en nuestro programa. Lo bueno de esta herramienta, es que podemos visualizar qué valores tienen las variables al entrar en un *Card* determinado.

FORMATEO DEL DECK

Otra curiosidad, es que si pulsamos en la pestaña *WML Deck,* podremos ver nuestro programa con el texto formateado. Esto podrá comprobarse en las imágenes que ilustran estas páginas.

Hay que tener en cuenta que si estamos en modo *WML Deck,* no podemos compilar el programa.

C:\Revista Solopro\fascículos de Wap\Fascículo 3\ejer2.wml

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/D
TD/wml_1.1.xml">
<wml>
<card id="card1" title="Primer Card" newcontext="true">
<p>
Card 1 ... <br/>
<!-- Las siguientes variables no se definirán hasta que entremos
en los otros Cards del Deck.
-->

card1 var1 = $(card1_var1) <br/>
card2 var1 = $(card2_var1) <br/>
card3 var1 = $(card3_var1) <br/>
<do type="accept" label="Siguiente Card">
<go href="#card2">
<setvar name="card1_var1" value="primero"/>
</go>
</do>
</p>
</card>
<card id="card2" title="Segundo Card">
<p>
Card 2 ... <br/>
```

Compile | Show | Line 40 | Col 40

Messages | Variables | History | Bookmarks | WML Deck | Session | ejer2.wml

Ready

Figura 7. - Nuestro programa escrito directamente y sin formateo.

EDITOR DEL ENTORNO DE DESARROLLO

Aunque en un artículo anterior dijimos que para realizar imágenes compatibles con *WML* podíamos utilizar un programa externo a *Nokia*, el entorno de desarrollo permite crear nuestras propias imágenes.

Para crear una nueva imagen sólo tenemos que seleccionar en el menú *File/New* la opción *WBMP Image*.

A continuación se muestran las posibles acciones que podemos realizar en el editor gráfico:

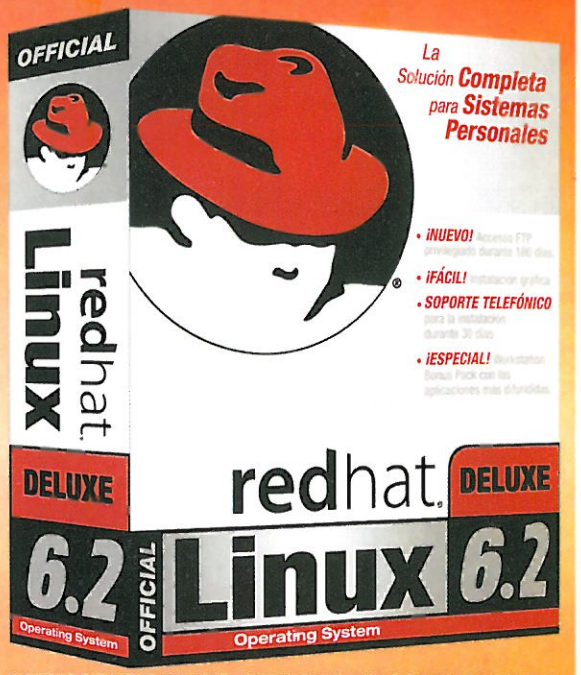
- Crear nuevos archivos *WBMP* de cualquier tamaño.
- Abrir archivos *WBMP* ya creados dentro del editor y la posibilidad de modificarlos.
- Importar *GIF* y *JPEG* y convertirlos a *WBMP*.
- Dibujar píxeles, líneas, cajas y elipses.
- Zoom de imágenes en el editor.
- Cortar, copiar, pegar y deshacer.
- Guardar archivos *WBMP*.
- Utilizar diversas herramientas para crear nuestras imágenes.
- Guardar la imagen mediante la opción del menú *File/ Save*.

El editor de WBMP del Toolkit permite crear nuestras propias imágenes

Para editar una imagen existente debemos realizar los siguientes pasos:

- Desde el menú *File*, seleccionamos *Open*. En ese momento aparecerá el diálogo *Open*.
- Seleccionamos desde el navegador el fichero que queremos editar y pulsamos en *Open*. El fichero se abrirá en el editor de imágenes.
- Editamos la imagen y la guardamos seleccionando *Save* en el menú *File*.

REDHAT LINUX
¡¡NUEVA VERSION 6.2 OFICIAL!!



STANDARD 7.400 pts. | DELUXE 14.900 pts. | PROFESIONAL 29.000 pts.

Precios de venta recomendados - IVA no incluido

Versión 6.2 completa y oficial para Europa

2 CD ROM's y cientos de aplicaciones

Manuales en castellano de instalación y referencia

Caja y presentación en castellano

Programa de instalación gráfica

escritorio basado en GNOME o KDE

Soporte telefónico, por e-mail, FTP y Web según versiones

MAYORISTA OFICIAL redhat PARA ESPAÑA

Precios especiales a distribuidores, tiendas e integradores según cantidades
¡Consúltenos!

¡¡Visite nuestro Web!!
http://www.abcnet.es/linux

abc analog, s.l.
1970-2000 30 años
91 634 20 00
FAX 91 634 47 86

Programación OpenGL en Delphi (I)

Juan Luis Ceada Ramos.
Diplomado en Informática de Sistemas.
Emilio Nestal Díaz.
Ingeniero Técnico en Informática.

Este mes veremos cómo crear objetos complejos a partir de otros más simples, así como la manera de optimizar nuestros programas para que la visualización de los objetos sea más eficiente.

CUÁDRICAS

Las cuádricas son las figuras geométricas que podemos dibujar en *OpenGL* con las funciones que proporciona la librería *GLU*, y que básicamente se refieren a conos, cilindros, discos y esferas.

VARIABLES DE ESTADO

En primer lugar, tendremos que crear una nueva variable de estado. Dicha variable almacenará el estado actual de una cuádrica: su estilo de dibujo, su orientación, su iluminación, etcétera. Para ello, basta con hacer:

```
Var Cuádrica: PGLUquadricObj;
Cuádrica := gluNewQuadric;
```

El tipo de datos *PGLUquadricObj* se encuentra definido en el fichero *OpenGL.pas*. El estado de una cuádrica no incluye la forma geométrica que se va a dibujar. Es decir, todas las formas geométricas tienen los mismos parámetros: estilo de dibujo, orientación, etcétera.

Si vamos a dibujar una escena donde todas las cuádricas tengan un aspecto parecido, basta con crear una única variable de estado, que se pasará como parámetro a las distintas funciones de creación de cuádricas.

CAMBIOS DE ESTADO

En segundo lugar, tendremos que cambiar el estado de la cuádrica. Es decir, ya tenemos la variable de estado creada, pero tendremos que asignarle los valores adecuados, de forma que cada cuádrica pueda personalizar su visualización. Para ello existen varias funciones:

- *GluQuadricDrawStyle(estado: PGLUquadric; Estilodibujo:GLEnum)*
- *gluQuadricNormals(estado: PGLUquadric; Normales:GLEnum)*
- *gluQuadricOrientation(estado: PGLUquadric; Orientacin:GLEnum)*

- *gluQuadricTexture(estado: PGLUquadric;Textura: GLEnum)*

Por supuesto, al ser creadas las variables de estado, se inician con unos valores por defecto, que si nos convienen, nos permitirán omitir este segundo paso.

Las cuádricas son básicamente cilindros y esferas

La función *GluQuadricDrawStyle* permite modificar el estilo de relleno de las cuádricas. En la Tabla 1 se pueden apreciar los parámetros que admite esta función. Por defecto, el estilo de dibujado es *GLU_FILL*.

La función *GluQuadricNormals* permite modificar el tipo de cálculo usado para las normales de iluminación. En la Tabla 1 se pueden ver los parámetros que admite esta función. Por defecto, no se calculan normales de iluminación (*GLU_NONE*).

La función *gluQuadricOrientation* controla la dirección de dichas normales. Pueden ir hacia fuera (*GLU_OUTSIDE*), o hacia dentro de la cuádrica (*GLU_INSIDE*). De la dirección de dichas normales depende que podamos ver (o no) las figuras.

La función *gluQuadricTexture* activa (*GL_TRUE*) o desactiva (*GL_FALSE*) la generación de coordenadas de textura.

Sobre la iluminación de las cuádricas y las texturas, volveremos a hablar en próximas entregas.

DIBUJOS DE CUÁDRICAS

Por ultimo, sólo nos queda dibujar las figuras que deseemos. Como dijimos, podremos dibujar cilindros, esferas, conos y discos.

Los cilindros se dibujan con la función *gluCylinder*. Los cilindros no tienen tapadas las partes superior ni inferior. Si por ejemplo queremos dibujar una tubería, basta con dibujar un cilindro. Si queremos hacer un tambor, tendríamos que dibujar un cilindro y dos discos que lo cierren.

Los parámetros que admite *gluCylinder* son los siguientes:

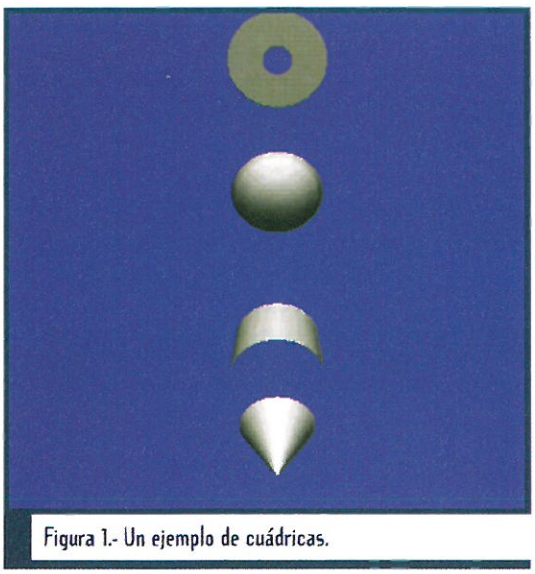


Figura 1.- Un ejemplo de cuádricas.

- Variable de estado.
- Radio de la base: *GL Double* (indica el radio que tendrá la circunferencia de la base del cilindro).
- Radio superior: *GLDou ble* (indica el radio que tendrá la circunferencia de la parte superior del cilindro.)
- Altura: *GLDOuble* (indica la longitud del cilindro.)
- Segmentos: *GLInt* y Pilas: *GLInt* (controlan el número de subdivisiones generadas a lo largo y alto del cilindro.)

Los dos últimos parámetros se encargan de controlar el realismo que tendrá el cilindro, mientras que los segmentos controlan su redondez. Las pilas controlan aspectos relativos a la iluminación. A más pilas, mayor calidad de reflejos, luces, etc. Se puede ver una muestra en la Figura 2.

Con respecto a los conos, no tenemos ninguna función especifica para dibujarlos, puesto que un cono no es más que un cilindro, sólo que la circunferencia de la base (o la superior) tiene un radio de tamaño 0.

Un cono se obtiene como un cilindro

Con respecto a los discos, hay que decir que son formas redondas y planas que pueden contener un agujero en el interior (ejemplo, las monedas de 25 pesetas, un anillo, etcétera.) Se dibujan con la función *gluDisk*, que recibe los siguientes parámetros:

- Variable de estado.
- Radio interior: *GLDouble* (controla el tamaño del agujero.)
- Radio exterior: *GLDouble* (controla el tamaño del disco.)
- Segmentos: *GLInt* (caras del disco.)
- Vueltas: *GLInt* (controla el número de anillos concéntricos que se dibujan en el interior del disco.)

Al igual que en los cilindros, los dos últimos parámetros controlan la calidad de visualización e iluminación (ver Figura 3.)

También es posible dibujar un disco parcial (ejemplo, *sprite* del protagonista del juego *pac-man*).

TABLA 1. Parámetros de las funciones de cambio de estado	
gluQuadricDrawStyle	
GLU_FILL	Las cuádricas se dibujan sólidas, usando polígonos
GLU_LINE	Las cuádricas se dibujan a base de líneas (dibujo en alambre)
GLU_SILHOUETTE	Las cuádricas se dibujan a base de líneas (dibujo en alambre), aunque sólo se dibuja el perfil de los polígonos.
GLU_POINT	Las cuádricas se dibujan usando puntos.
gluQuadricDrawStyle	
GLU_NONE	No se calculan las normales de iluminación
GLU_FLAT	Se calculan normales de iluminación para cada polígono
GLU_SMOOTH	Se calculan normales de iluminación para cada vértice del polígono, mejorando la iluminación

Para ello, usaremos la primitiva *gluPartialDisk*, idéntica a *gluDisk*, pero con dos parámetros más, que definen el ángulo de inicio y fin del disco (ver Figura 4).

Es posible crear discos mediante cilindros de altura 0

Con respecto a las esferas, la función que las dibuja es *gluSphere*, (ver Figura 5), que recibe los siguientes parámetros:

- Variable de estado.
- Radio: *GLDouble* (controla el tamaño de la esfera).
- Segmentos: *GLInt*.
- Pilas: *GLInt*.

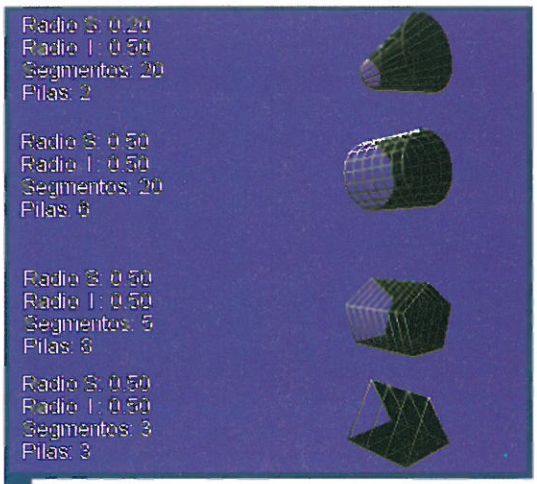


Figura 2.- Cilindros con diferentes parámetros.

En todos los casos, las cuádricas se dibujan desde las coordenadas *(0,0,0)*, por lo que tendremos que desplazarnos antes de dibujar cualquier figura. Es decir, si queremos dibujar un disco en *(5,4-10)* de radio *5*, tendremos que hacer:

```
GlTranslate(5,4,-10)
gluDisk(Estado, 0,5,15, 5)
```

Nos ayudaremos de las pilas de matrices para enlazar objetos

Para poder enlazar unas figuras con otras tendremos que hacer operaciones con las coordenadas, ayudados por las pilas de matrices.

En el ejemplo que veremos más adelante se siguen los siguientes pasos para dibujar un lápiz:

- Nos movemos a una determinada coordenada.
- Guardamos la posición en la pila.
- Dibujamos el cilindro.
- Cambiamos de posición.
- Dibujamos la punta del lápiz (un cono.)
- Sacamos la posición de la pila.
- Dibujamos el disco de la base. Este disco se dibuja en el mismo origen que el cilindro, puesto que hemos restaurado la matriz de modelado. El disco coincide completamente con la base del cilindro, "cerrándolo".

Pero lo que realmente nos hará expertos en enlazar objetos simples es la experiencia, que obtendremos a lo largo de muchas pruebas.

LISTAS DE VISUALIZACIÓN

Las listas de visualización son utilizadas para reducir el número de cálculos necesarios para dibujar las figuras, y de esta forma, mejorar la velocidad de dibujado. Al crear la lista de visualización, se hacen una serie de cálculos (es como si se compilasen todas las funciones de dentro de la lista.) Los resultados de estos cálculos se almacenan, de forma que en sucesivas llamadas a la lista de visualización se aprovechan, evitando que tengamos que volverlos a calcular.

Una lista de visualización reduce los cálculos que se deben realizar

Por regla general, hacer *X* operaciones por separado es más lento que realizarlas dentro de una lista de visualización. En la práctica, la diferencia sólo se nota cuando el número de polígonos de la escena es considerable (cientos o miles.)

Las listas se crean mediante las funciones *glNewList* y *glEndList*. Estas dos funciones, delimitan qué otras funciones de *OpenGL* se van a incluir dentro de la lista. La función

LISTADO 1. Ejemplo del uso de listas de visualización

```
procedure TFPrincipal.FormPaint
 (Sender: TObject);
 begin
  ...
  glCallList(Lapiz);
  ...
  SwapBuffers(Canvas.Handle);
  wglMakeCurrent(0, 0);
end;

function CreaObjeto: Cardinal;
var
  Quadric: PGLUquadricObj;
begin
  Result := glGenLists(1);
  glNewList(Result, GL_COMPILE);
    Quadric := gluNewQuadric;
    GluQuadricDrawStyle
     (Quadric, GLU_FILL);
    gluQuadricNormals
     (Quadric, GLU_SMOOTH);
    gluQuadricOrientation
     (Quadric, GLU_OUTSIDE);
    glTranslatef(0,0,-2);
    gluCylinder(Quadric, 0.25,
     0.25, 3, 6, 2);
    glPushMatrix;
    glTranslatef(0, 0, 3);
    gluCylinder(Quadric, 0.25,
     0, 0.5, 10, 2);
    glPopMatrix;
    gluQuadricOrientation
     (Quadric, GLU_INSIDE);
    gluDisk(Quadric, 0, 0.25,
     6, 2);
    gluDeleteQuadric(Quadric);
  glEndList;
  if not Reconfigura then Halt;
end;

procedure TFPrincipal.
   Configuracion;
begin
  ...
  Lapiz := CreaObjeto;
  ...
end;
```

glNewList tiene dos parámetros: el primero es un número entero que identifica a la lista. El segundo indica si *OpenGL* debe compilar la lista, dejándola preparada para usos futuros (*GL_COMPILE*) o compilarla y ejecutarla inmediatamente (*GL_COMPILE_EXECUTE*).

OpenGL optimiza los cálculos compilando las funciones del interior de una lista

Después, para usar la lista que hemos creado sólo tenemos que llamarla mediante la función *glCallList*, a la que le pasaremos el número de la lista que queramos ejecutar. Nada nos impide llamar a otras listas desde dentro de una lista (hasta 64 niveles de anidamiento.) Incluso es posible llamar a varias listas sucesivamente. Para ello, crearemos una matriz de enteros, donde almacenaremos los identificadores de las listas de uso. Después, mediante *glCallLists*, usaremos las listas. Por ejemplo:

```
Var MisListas: Array [1..5] of
    GLInt;
....
MisListas[1]:=ID_LISTA_1;
....
MisListas[5]:=ID_LISTA_5;
GlCallLists(5, GL_INT, @Mis
    Listas);
```

El primer parámetro de *glCallLists* indica el número de elementos de la matriz. El segundo, indica el tipo de datos almacenados en la matriz. El tercero es un puntero a la matriz de datos (que debe ser del tipo indicado en el segundo parámetro.)

Con *glDeleteLists(lista: cardinal, rango:integer)*, podemos borrar las listas de visualización que deseemos. Por ejemplo, *glDeleteLists(1,50)* borra todas las listas cuyo *ID* se encuentre entre el 1 y el 50.

Es posible compartir una variable de estado entre diferentes cuádricas

Mediante *glIsList(lista:cardinal)* podemos averiguar si un *ID* ya ha sido asignado a una lista. De esta forma, nos evitamos los problemas derivados de asignar el mismo identificador a diferentes listas (en este caso, la última machacaría a la anterior.) Esta función devuelve *GL_TRUE* en caso de que el *ID* que le pasemos ya esté asignado.

EN LA PRÁCTICA

En esta serie de ejemplos, veremos en primer lugar, cómo realizar modelos complejos a partir de las primitivas básicas de *OpenGL*, concretamente usaremos la suministradas en la librería *GLU*.

En el segundo, añadiremos el uso de las listas de visualización,

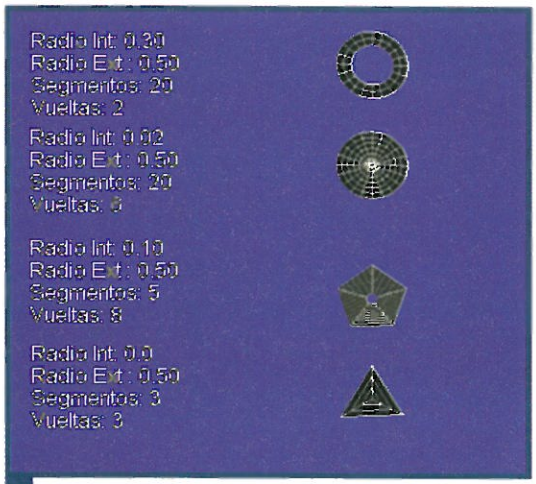


Figura 3.- Discos con diferentes parámetros.

Figura 4.- Discos parciales, con ángulo de inicio 0 y de fin 300.

para posteriormente comentar las ventajas y los inconvenientes de su uso. Por falta de espacio, no mostramos el código fuente de los ejemplos en este artículo. Todo el código se encuentra en el *CD-ROM* profusamente comentado, por lo que no creemos que existan problemas para seguirlo.

MODELADO AVANZADO

Como hemos visto en las anteriores entregas, construir cualquier objeto a base de definir sus vértices es bastante complejo, ya que teníamos que definir no sólo los vértices sino la forma de unirlos.

OpenGL permite usar una serie de primitivas pertenecientes a la librería *GLU* para realizar objetos simples, los cuales podremos unir de forma jerárquica. A pesar de estas ventajas, no podemos olvidar lo visto hasta ahora, ya que alguna vez tendremos que recurrir de nuevo a ellas.

En este caso hemos realizado un objeto complejo, concretamente un lápiz. No vamos a definir sus vértices, sino que vamos a usar algunas de las primitivas que hemos visto para dibujarlo. Concretamente, el cuerpo del lápiz, esta formado por un prisma de seis caras, construido a partir de la primitiva *gluCylinder*.

La parte de la punta está construida también con la primitiva *gluCylinder*, pero en este caso, lo que hemos modificado ha sido el valor del radio superior, que al definirlo como 0, producirá cono. Por tanto, a partir de esta primitiva simple, podemos construir todo tipo de conos, prismas, etcétera.

La parte inferior del lápiz, está construida con la primitiva *gluDisk*, que crea un polígono de tantas caras como le indicamos, y permite dejar un hueco en el centro de dicho polígono. Así, con la primitiva *gluDisk*, podemos crear desde triángulos hasta círculos, basta con definir el número de lados que deseemos.

La librería GLU suministra primitivas para construir objetos más complejos

Hay que destacar que ambas primitivas permiten asignar el número de divisiones horizontales que se definen a los planos. Aunque parezca un parámetro inútil, influye de forma considerable en el sistema de iluminación, al reducir el número de planos y vértices sobre los que realizar el cálculo de las normales.

Estos detalles, los veremos en próximas entregas, cuando veamos detenidamente el sistema de iluminación de *OpenGL*, pero podemos avanzar que un número alto de estas divisiones provoca efectos de luz sobre el objeto mucho más reales. Por el contrario, aumenta el numero de cálculos, tanto si utilizamos *GLU_FLAT* como con *GLU_SMOOTH*.

La aplicación de ejemplo se ha preparado para comprobar rápidamente las posibilidades de las cuádricas, y sobre todo, el estilo de dibujo y el tipo de cálculo de las normales.

Es posible controlar la calidad de dibujo e iluminación de las cuádricas

En el primer caso, podemos dibujar las primitivas mediante puntos, líneas, mediante la silueta, o bien con relleno completo. La diferencia entre dibujar con líneas o dibujar la silueta, es que la segunda no dibuja las líneas que separan los planos coplanarios, es decir, planos contiguos y con la misma normal.

En el segundo caso, podemos elegir el tipo de cálculo de las normales. En este caso tenemos tres

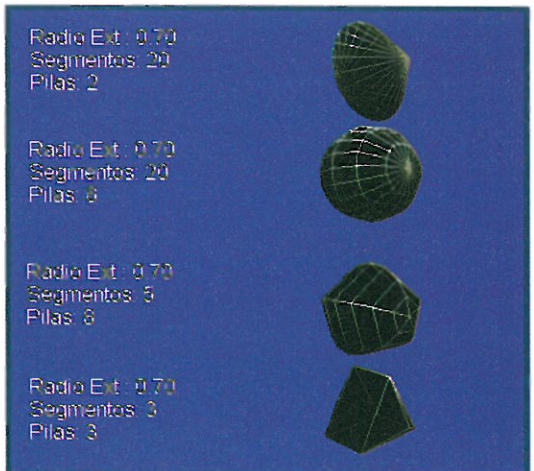


Figura 5.- Esferas con diferentes parámetros.

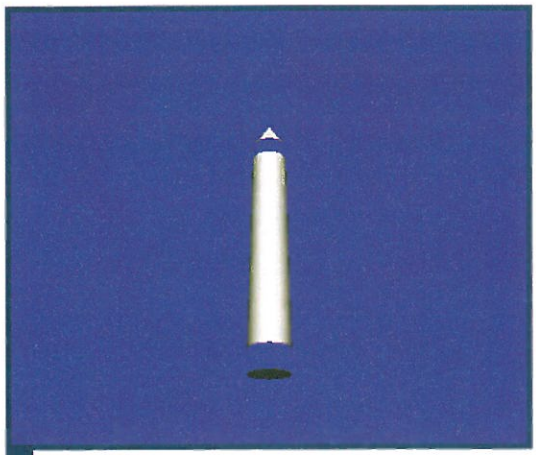


Figura 6.- Una nuestra del enlace de objetos.

opciones: *GLU_NONE*, con la que no se calculan las normales para la iluminación, por lo que el objeto no queda afectado por la iluminación. *GLU_FLAT* donde se calcula una normal por plano y centrada en él. Y por último, *GLU_SMOOTH* que provoca el cálculo de las normales de iluminación en cada vértice, consiguiendo un aspecto más real, pero más costoso en tiempo de ejecución.

Existe otra característica, la orientación de las cuádricas, que permite definir la dirección de la normal que orienta cada plano. En el ejemplo, hemos utilizado las dos posibilidades, *GLU_OUTSIDE*, para el prisma y el cono, y *GLU_INSIDE*, para la parte posterior, ya que en caso contrario dicha parte no se vería, al quedar la normal apuntando hacia el interior del prisma principal. Sólo la veríamos si estuviésemos situados dentro del cuerpo del lápiz. Como norma general, las normales deben apuntar siempre hacia el exterior del objeto.

Tanto el estilo de dibujo de las cuádricas como el cálculo de las normales se pueden utilizar con diferentes parámetros en el mismo objeto. Podemos construir un objeto donde parte de él se vea como líneas y sin cálculo de normales, y la otra se rellene completamente y se calcule las normales en cada vértice. Esta posibilidad la podéis comprobar en tiempo de ejecución en el primer ejemplo, ctivando/desactivando los *checkboxes*.

LISTAS DE VISUALIZACIÓN

En el segundo ejemplo, vamos a utilizar las listas de visualización, que nos van a permitir crear un código muy veloz, a costa de perder flexibilidad en las posibilidades de modificación de los objetos en tiempo real.

Como ya se vio anteriormente, las listas de visualización se deben crear antes de ser utilizadas. En este caso, hemos utilizado una lista para dibujar el lápiz del ejemplo anterior, la cual es creada en el procedimiento encargado de la configuración del sistema.

Las variables de estado permiten modificar la apariencia de las cuádricas

Una vez creada la lista de visualización, la podemos utilizar cuando deseemos, concretamente hemos sustituido el código que dibujaba el lápiz, por la llamada a la lista que lo dibujará.

El código resultante es más sencillo de leer, sobre todo cuando crece, y facilita el uso de objetos que se repiten en la escena. Con ello, se consigue también que nuestra implementación *OpenGL* se encargue de optimar temporalmente dichas instrucciones, llegando hasta una mejora del 100% si el número de triángulos que dibujar es lo suficientemente grande. Esta situación se dará cuando veamos las superficies *NURBS*, donde los triángulos que forman la escena se pueden contar por decenas de miles.

Un inconveniente del uso de las listas de visualización, es que al crear las listas antes de empezar la ejecución normal del programa, tendremos que ejecutar de nuevo la creación de esa lista, con el consiguiente gasto de tiempo.

Las listas se pueden anidar hasta una profundidad de 64 llamadas

Para evitar este problema, podemos dividir nuestros objetos en partes pequeñas, que sean fácilmente recalculadas. Así, conseguimos atenuar el problema de las modificaciones en tiempo real y continuar con las ventajas de su uso.

El uso de las listas de visualización en la práctica es muy sencillo, lo único que necesitamos en una variable de tipo entero que almacene el identificador de la lista. Utilizar dicho identificador para crear la lista, y colocar las instrucciones que necesitemos entre las llamadas a *glNewList* y *glEndList*, y cuando necesitemos dichas instrucciones, simplemente llamaremos a *glCallList*; tan simple como eso.

Recordaremos que las listas de visualización se pueden anidar, para construir objetos complejos, siempre y cuando no se rebase el límite de 64 llamadas.

Con todo esto, hemos incluido en la escena varios objetos, en este caso varios lápices como los del primer ejemplo. Como podréis ver, el código es más simple, y si tuviese un número alto de triángulos, comprobaríamos su eficiencia temporal.

¿TE FALTA ALGÚN NÚMERO DE...?

LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO

SÓLO PROGRAMADORES

ATENCIÓN

Los números 49 al 51, ambos inclusive,

se encuentran AGOTADOS

BOLETÍN DE PEDIDOS

Rellene o fotocopie el cupón y envíelo a REVISTAS PROFESIONALES, S.L.
(Revista Sólo Programadores). C/ San Sotero, 5. 1ª Planta. 28037 Madrid
Tlf: 91 304 87 64. Fax: 91 327 13 03

Deseo que me manden los número/s.....
.....
.....

NOMBRE Y APELLIDOS:.....
.....

EDAD:..... PROFESIÓN: .....TFNO: .....
DOMICILIO: .....
CIUDAD: .....
PROVINCIA: .....C.P: .....

❑ **Precio por unidad: 975 pesetas** + 700 ptas. de gastos de envío.

FORMAS DE PAGO:

❑ Giro postal a nombre de REVISTAS PROFESIONALES, S.L
❑ Talón bancario a nombre de REVISTAS PROFESIONALES. S.L
❑ Domiciliación bancaria
❑ Contra reembolso

DATOS DE DOMICILIACIÓN:

Banco.....
Domicilio.....
Nº de Cuenta.....|.....|.....|.....
Titular.....
Fecha.....

FIRMA

Programación orientada a objetos (III)

Jordi Agost Moré.
Profesor de Programación Multimedia de la Universidad de Lleida.

Seguimos explorando en este artículo todas las posibilidades que ofrece *Visual Basic 6* en la programación orientada a objetos.

OPERACIONES CON OBJETOS

En el artículo anterior comenzamos a introducirnos en la programación de objetos en *Visual Basic,* estudiando las características de los mismos a través de las clases, que son las que van a definir las propiedades, atributos y métodos de un objeto dentro de su grupo.

Así, vimos la forma de obtener información de las clases, para conocer las propiedades de los objetos, además de aprender a crear otras para añadirlas a nuestro proyecto, definiendo sus propiedades, y creando los métodos de comportamiento y los eventos *Initialize* y *Terminate.*

Tras ese primer estudio, en este número vamos a estudiar con más detenimiento las características de las clases *Visual Basic 6,* para centrarnos más en el propio objeto.

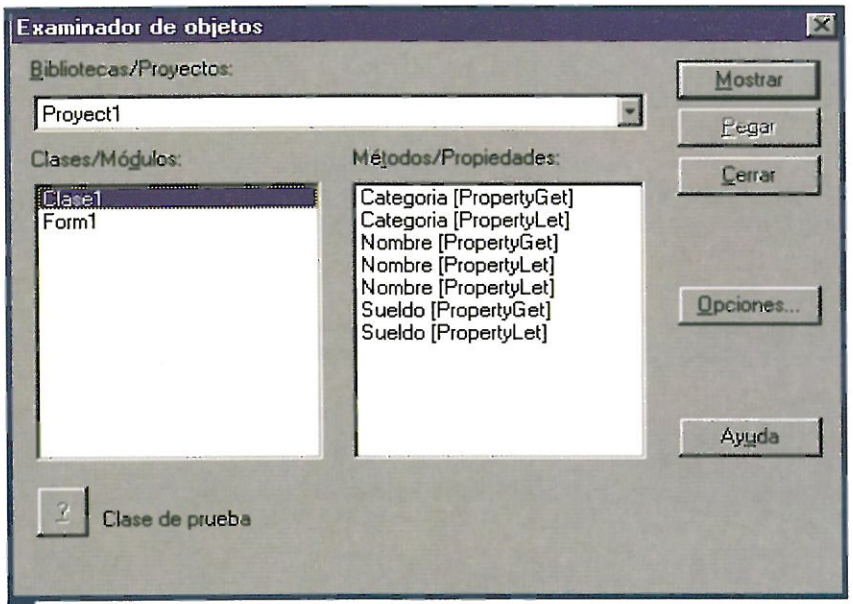


Figura 1.- Todas nuestras clases aparecerán como objetos del Examinador de objetos.

Figura 2.- Diagrama simbólico, donde se aprecia la deriva de instancias de nuestra clase a partir de una clase base.

En efecto, son muchas las operaciones que podemos hacer con los objetos, el auténtico protagonista de este programa. Seguidamente, vamos a estudiar algunas formas de optimar su gestión.

Casi todos los objetos en Visual Basic tienen propiedades predeterminadas

Si lo que deseamos es efectuar diferentes acciones sobre el mismo objeto, como por ejemplo establecer diferentes propiedades, el proceso que haríamos normalmente es el siguiente:

```
Private Sub Form_Load()
    Command1.Caption = "Prueba"
    Command1.Visible = True
    Command1.Enabled = True
End Sub
```

No obstante, hay una forma más práctica de obtener el mismo resultado:

```
Private Sub Form_Load()
    With Command1
.Caption = "Prueba"
     .Visible = True
     .Enabled = True
    End With
End Sub
```

PROPIEDADES PREDEFINIDAS

Otra característica propia de los objetos, consiste en que muchos de ellos tienen propiedades predeterminadas, cuyo fin es simplificar el código haciéndolo más legible.

Por ejemplo, si tenemos un control de cuadro de texto llamado *Text1* y escribimos:

```
Text1.Text = "Prueba"
```

Sería equivalente escribir:

```
Text1 = "Prueba"
```

Elijamos una forma u otra, lo realmente importante es que seamos siempre congruentes escribiendo el código.

También podemos utilizar las propiedades por defecto si trabajamos con una variable de objeto y ésta almacena una referencia a un objeto dado. Este aspecto se puede observar en el código siguiente:

```
Private Sub Command1_Click()
    Dim obj as Object
    Set obj = Text1
    Obj = "Prueba"
End Sub
```

Sin embargo, debemos ir con cierto cuidado cuando utilicemos variables de tipo *Variant* para proceder como hemos indicado anteriormente, ya que *Visual Basic* trata de forma especial este tipo de variables, debido a que pueden contener datos de muchos tipos diferentes. Así, es posible leer el valor de una propiedad predeterminada, pero no establecerlo. Por ejemplo si dentro de un procedimiento declaramos lo siguiente:

```
Dim v As Variant
```

Y establecemos la propiedad determinada de *Text* a *Prueba*

```
Text1 = "Prueba"
```

Asignamos la variable y la mostramos

```
Set v = Text1
MsgBox v
```

Pero si hacemos:

```
v  = "Prueba2"
MsgBox v
```

El último *MsgBox* dará como resultado la cadena *Prueba2,* y *Text1* no habrá cambiado.

GUARDANDO DATOS DE CLASE

Con lo aprendido hasta ahora, nos encontramos con el problema de que en el momento en que todas las referencias a un objeto han desaparecido, dicho objeto se destruye, y con él todos los datos que poseía.

Para evitar que suceda esto, nos veremos obligados a grabar los datos en un fichero o en una base de datos. La mejor forma para llevar a cabo dicho proceso, es crear una clase genérica que se encargue de grabar los diferentes objetos, o de leerlos, según convenga.

En cualquier caso, si construimos una clase genérica, tendremos el problema de que dicha clase no conocerá los elementos que deberán leerse o grabarse de cada objeto (éstos pueden ser muy diferentes y tener tratamientos distintos, por ejemplo, *strings*, *integers*...), por lo que cada objeto tendrá que tener sus propios métodos para grabar sus datos.

Entonces, podemos preguntarnos: ¿cuáles son las ventajas de tener una clase genérica? Principalmente, podemos responder, que la claridad y la abstracción de programación.

Utilizaremos clases genéricas para ganar en claridad y abstracción de programación

Por ejemplo, sólo tendremos un lugar en el código donde abriremos y cerraremos los ficheros. Sólo habrá un lugar donde comprobaremos errores tales como de entrada y salida, si el fichero existe, si está siendo usado, etcétera.

Además, podríamos cambiar en un determinado momento la forma en que se lee un fichero sin cambiar apenas código en la aplicación (sólo en el método del objeto afectado), ya que siempre que quisiéramos grabar o leer algo, lo haríamos mediante esta clase general.

Suponiendo que deseamos crear una clase general para grabar nuestros datos en un fichero, crearemos una clase genérica que se llamará *Cfichero* con las siguientes variables miembro:

```
Option Explicit

Private m_NombreFichero As String
Private m_NumFichero As Integer
```

Cuando desaparecen todas las referencias a un objeto, dicho objeto se destruye

Una propiedad necesaria, será la del nombre del fichero. Dicha propiedad ha de ser de lectura y escritura, por lo que implementaremos los dos procedimientos de propiedades (*Let* y *Get*). Su implementación sería la siguiente:

```
Public Property Let Nombre(vNew-
    Value)
m_NombreFichero = Nombre
End Property

Public Property Get Nombre()
Nombre = m_NombreFichero
End Property
```

Además, necesitaremos un método que escriba el objeto en el fichero de datos que hayamos elegido, según lo indicado en la Tabla 1:

TABLA 1

```
Public Sub EscribeObjeto(obj As Object)

Kill m_NombreFichero
m_NumFichero = FreeFile
Open m_NombreFichero For Binary As #m_NumFichero
Select Case Err.Number
  Case 0 ' no hay errores.
    obj.EscribeDatos
  Case Else
    'tratamiento de errores
  End Select
Close m_NumFichero
End Sub

Public Sub LeeObjeto(obj As Object)
m_NumFichero = FreeFile
Open m_NombreFichero For Binary Access Read As #m_NumFichero
Select Case Err.Number
  Case 0 ' no hay errores
    Do Until EOF(m_NumFichero)
      obj.LeerDatos
    Loop
  Case Else
      ' tratamiento de los diversos errores
  End Select
Close #m_NumFichero
End Sub
```

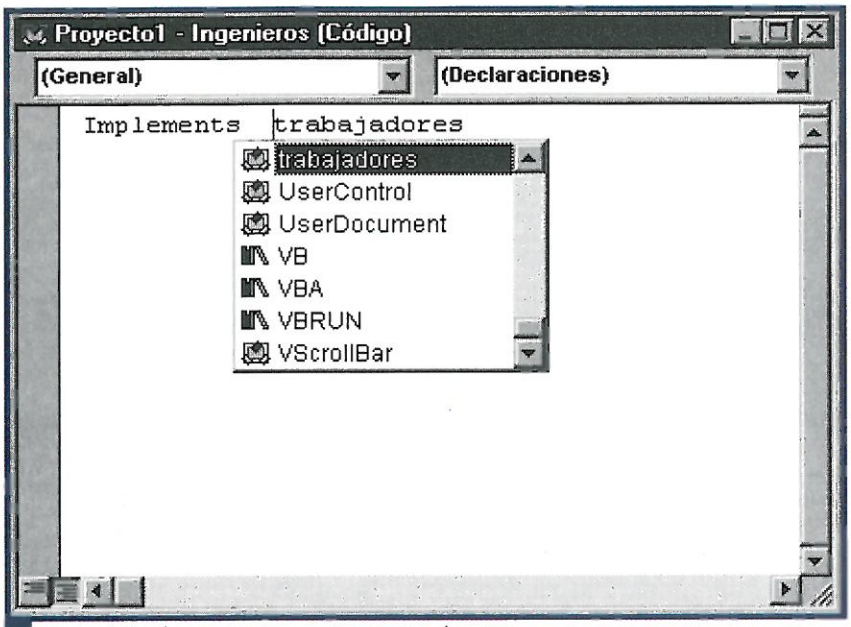


Figura 3.- La clase trabajadores se ha añadido a los objetos existentes.

IMPLEMENTACIÓN DE UNA INTERFAZ

Tal y como hemos dicho, una interfaz será simplemente un conjunto de propiedades y métodos. Ahora vamos a crear una interfaz llamada *trabajadores,* y la implementaremos en dos clases: *Ingenieros* y *obreros*. Creamos la interfaz *trabajadores* agregando un módulo de clase al proyecto, e insertamos el siguiente código:

```
Public Sub Trabajar (ByVal horas
    as integer)

End Sub

Public Sub MoverDepartamento
    (ByVal Ob as Object)

End Sub
```

Destaquemos que dichos métodos no contienen código, ya que *trabajadores* es una clase abstracta que no contiene código de implementación. El objetivo de dicha clase, es crear una plantilla de interfaz que añadiremos a otras clases.

Acto seguido, vamos a agregar dos módulos de clase más, un primer módulo llamado *ingenieros* y otro *obreros*, con los que implementaremos la interfaz *trabajadores*. Este paso lo podremos efectuar con la instrucción *Implements* al principio del módulo:

```
Option Explicit
Implements trabajadores
```

Una vez realizado dicho paso, veremos cómo aparece una lista desplegable de los objetos. A continuación podremos hacer clic para seleccionar *trabajadores* de la lista.

Acto seguido, si miramos los objetos en la ventana de código, en la lista desplegable de la derecha aparecerán los procedimientos de la interfaz *trabajadores.*

Como nota importante, cabe destacar que al implementar una interfaz, la clase tiene que responder a cualquier propiedad o método de la misma. Por lo tanto, sacamos la conclusión de que una clase debe implementar todas las propiedades y métodos de una interfaz.

Ahora podríamos incluir todo el código que queramos en la clase *ingenieros:*

```
Private Sub trabajadores_MoverDe-
    partamento(ByVal Ob As
    Object)
Debug.Print "Ingeniero moviendo
    departamento"
End Sub

Private Sub trabajadores_Traba-
    jar(ByVal horas As Integer)
Debug.Print "Ingeniero trabajan-
    do"
End Sub
```

Ahora, en este punto, la clase *ingenieros* posee dos interfaces: la interfaz *trabajadores* que hemos implementado, y que posee dos miembros, y la interfaz *ingenieros* predeterminada, que por ahora no tiene miembros.

Una interfaz será simplemente un conjunto de propiedades y métodos

Para ver un ejemplo más claro de cómo implementar estos métodos dentro de un programa cualquiera, imaginemos que tenemos un formulario, y en el método *Form_Load* añadimos dicho código:

```
Private Sub Form_Load()

Dim ing As Ingenieros
Dim obr As obreros
Dim tra As trabajadores
Set ing = New Ingenieros
Set obr = New obreros

Set tra = ing
```

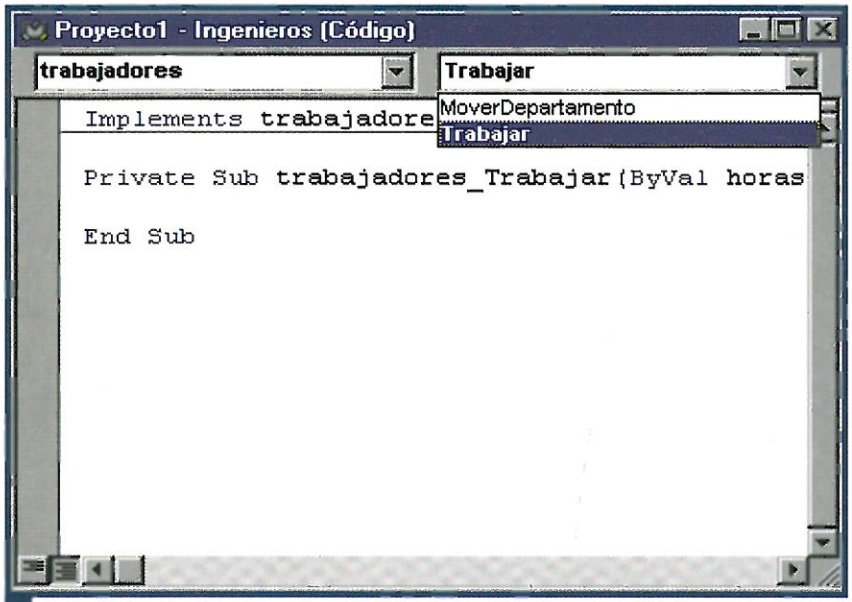


Figura 4.- La clase ha adquirido los procedimientos de la interfaz trabajadores.

```
Call tra.MoverDepartamento(obr)

Set tra = obr
Call tra.MoverDepartamento(obr)

End Sub
```

Como se puede apreciar, en primer lugar declaramos tres variables: una de tipo *ingenieros*, una de tipo *obreros* y otra de tipo *trabajadores*. Asignamos las variables de tipo *ingenieros* y *obreros* a dos objetos nuevos. A continuación, la variable *tra* (de tipo *trabajadores*), que puede tener cualquier referencia a cualquier objeto que implemente la interfaz *trabajadores,* la asignamos a la variable *ingenieros* en principio, para así llamar al método de *MoverDepartamento*.

Si ahora, en la clase *trabajadores* queremos incluir una propiedad que contenga la *edad,* lo podemos hacer mediante una variable de tipo *Public*, en la sección de declaraciones:

```
Option Explicit
Public Edad as Integer
```

En este momento, las listas desplegables de procedimiento de los módulos de las clases *ingenieros* y *obreros,* tienen los procedimientos requeridos para implementar la propiedad *Edad*.

Una clase siempre tiene que implementar todos los métodos y propiedades de una interfaz

Debemos decir, que el empleo de este método (la declaración de una propiedad mediante una variable *Public*) está pensado simplemente para la comodidad del programador, ya que internamente, *Visual Basic* implementa las propiedades como parejas de procedimientos de tipo propiedad.

Si por ejemplo, quisiéramos poner un límite de edades, podríamos poner dicho código en el procedimiento *Let* de *trabajadores _Edad*. La única cosa que tenemos que tener en cuenta, es que la interfaz coincida con la descripción de la biblioteca de tipos (es decir que coincidan los tipos de las llamadas).

A continuación, hablaremos de las diferentes interfaces que puede tener un objeto. Para este supuesto vamos a examinar el siguiente código:

```
Private Sub Command1_Click()

Dim ing as ingeniero
Dim tra as trabajadores

Set ing = New ingenieros
Set tra = ing
MsgBox TypeName(tra)

End Sub
```

Contrariamente a lo que pueda parecer, la instrucción *MsgBox* presentará el mensaje *trabajadores*. De dicho ejemplo, vamos a aprender algunas cosas. En *Visual Basic,* cada clase tiene una interfaz predeterminada, con el mismo nombre que el de la clase. Por ejemplo, la clase *ingenieros* tiene una interfaz por defecto, llamada *_Ingenieros* (el subrayado indica que la interfaz se encuentra oculta en la biblioteca de tipos). Pero como dicha clase también implementa la interfaz trabajadores, decimos que tiene una segunda interfaz.

Hablamos de polimorfismo en Visual Basic, cuando muchas clases pueden proporcionar la misma propiedad o el mismo método

No obstante, al final, debajo de cada interfaz, el objeto continúa siendo *trabajadores*, que es lo que nos viene a demostrar el ejemplo visto. Cuando asignamos un objeto *ingenieros* a una variable de tipo *trabajadores*, entonces *Visual Basic* preguntará al objeto *ingenieros* si acepta la interfaz de *trabajadores* (a través del método llamado

QueryInterface o *QI*), si la respuesta es afirmativa, el objeto se asignará a la variable, y a través de dicha variable sólo podremos acceder a los métodos y propiedades de la interfaz. En cambio, si la respuesta es negativa, se producirá un error.

Pensemos ahora qué sucede si asignamos una referencia de objeto a una variable genérica de objeto:

```
Private Sub Command1_Click()

Dim ing as Ingenieros
Dim tra as Trabajadores
Dim obj as Object

Set ing = New Ingenieros
Set tra =  ing
Set obj = tra
MsgBox TypeName(obj)

End Sub
```

El resultado será *ingenieros*, así que tenemos que tener en cuenta que la interfaz a la que la variable tendrá acceso será la última interfaz asignada.

POLIMORFISMO

Vamos a tratar por último, el polimorfismo dentro del ámbito de *Visual Basic*. Polimorfismo dentro del ámbito del *Visual Basic* significa que muchas clases pueden proporcionar la misma propiedad o el mismo método, y el que llama no tiene por qué saber la clase a la que pertenece el objeto antes de llamarla.

Por ejemplo, si tenemos una clase que sea *ingenieros* y otra *obreros*, las dos podrían tener un método *trabajar*. Polimorfismo significa que podemos invocar el método *trabajar* sin saber si nos estamos refiriendo a un ingeniero o a un obrero (teniendo en cuenta que las dos clases tendrán trabajos diferentes). Logramos así una mayor claridad de código y una funcionalidad mayor, ya que podemos "olvidarnos" del objeto al cual hacemos referencia.

La mayoría de lenguajes de programación utilizan el polimorfismo mediante la herencia, es decir, en nuestro caso, las clase *ingenieros* y *obreros*, heredarían las características de una clase llamada *trabajadores*, además, cada clase (*ingenieros* y *obreros*) haría válido su método *trabajar*, invalidando de esta forma el método *trabajar* de la clase trabajadores.

Visual Basic no utiliza la herencia para proporcionar polimorfismo, la proporciona mediante lo que llamaríamos múltiples interfaces *ActiveX*. Una interfaz, es un conjunto de propiedades y métodos relacionados entre ellos. En *Visual Basic* creamos una interfaz llamada *trabajador*, y la implementamos en la clases *ingeniero* y *obrero*. Una vez hecho este paso, podremos invocar al método *trabajar* de cualquiera de las dos clases.

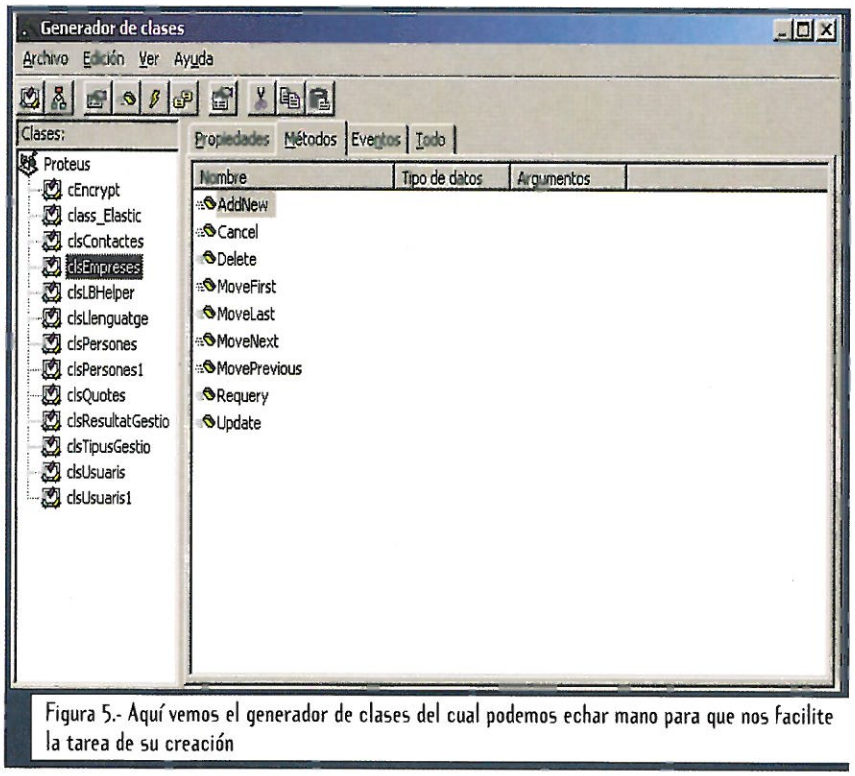


Figura 5.- Aquí vemos el generador de clases del cual podemos echar mano para que nos facilite la tarea de su creación

Visual Basic no utiliza la herencia para proporcionar el polimorfismo, haciéndolo con múltiples interfaces ActiveX

Hasta aquí, hemos hecho una primera aproximación a las principales herramientas que posee *Visual Basic* para tratar la programación orientada a objetos. Utilizando dichas herramientas adecuadamente, veremos cómo nuestros programas se vuelven mucho más claros y modulares, al tiempo que ganan en legibilidad y solidez. En el próximo artículo, desarrollaremos paso a paso diversas clases, para así descubrir su funcionalidad, y poderlas utilizar dentro de nuestros programas.

C++ Builder 5

Constantino Sánchez Ballesteros.
Técnico Superior de Desarrollo de Aplicaciones Informáticas.

C++ Builder 5 de *Borland* ya está disponible. En este artículo vamos a dar un recorrido por las novedades de este completo paquete de programación.

MENÚS

C++ *Builder* comienza una nueva andadura con esta quinta versión. A tenor de sus nuevas características se presenta como una alternativa muy seria a *Visual C++*. Por supuesto, asimilar el cambio de un entorno a otro cuesta algo de tiempo, pero puede merecer la pena, sobre todo por el carácter visual de *C++ Builder*. En este artículo se presentarán las novedades incluidas en esta versión, dejando la comparación con *Visual C++* para artículos posteriores.

El nuevo Builder incorpora numerosas implementaciones de gran utilidad

A continuación se resumen algunas de las novedades más importantes de *Builder*:

- *ADOExpress*: nuevos componentes que permiten el acceso a una amplia variedad de formatos de bases de datos utilizando la tecnología *ADO* de *Microsoft*. Con estos componentes, no es necesitamos utilizar *Borland Database Engine (BDE)* para el acceso a datos.
- Compilador: *Builder* ahora puede compilar en modo *background*. Esto significa que podemos seguir editando el código fuente de nuestro programa mientras la compilación esté en progreso. También se ha mejo-

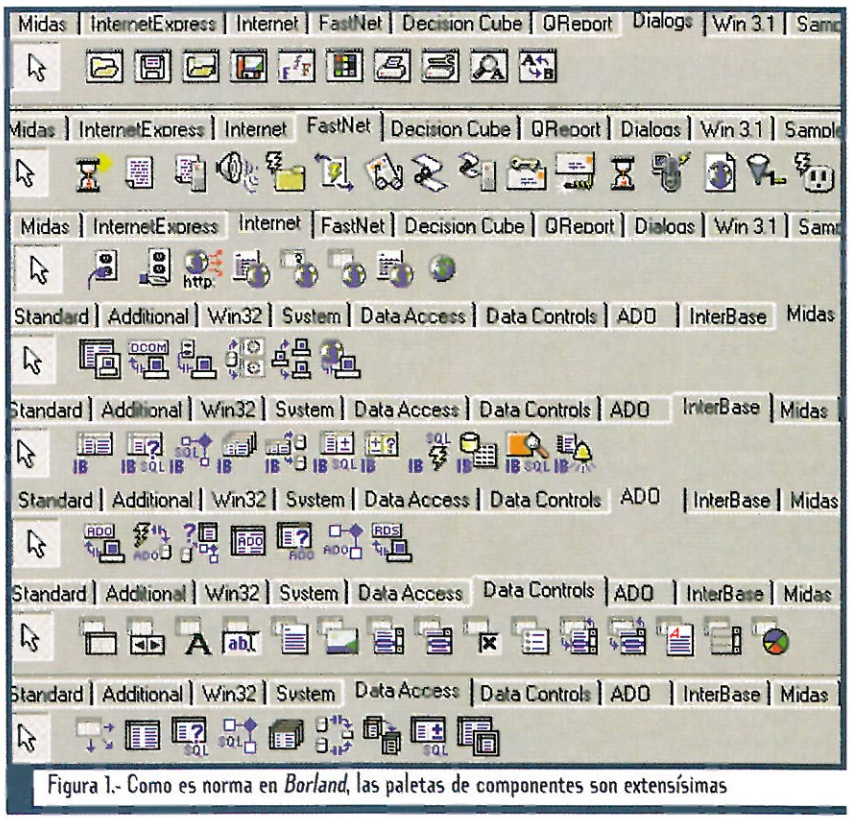


Figura 1.- Como es norma en *Borland*, las paletas de componentes son extensísimas

rado la compatibilidad con *Microsoft Visual C++*.

- *Linker*: nuevos *switches* para enlazar nuestros archivos objeto. Ahora también es posible crear *DLL's* que se carguen justo cuando sean llamadas desde el programa.

- *Debugger*: el depurador integrado tiene nuevas características, como establecer opciones de depurado para procesos específicos, nuevas opciones en el menú Run, etcétera.

- *COM+*: además de *COM*, en esta nueva versión se ha añadido soporte *COM+*, incluyendo nuevos asistentes para crear eventos *COM+* y nuevos objetos.

- Herramientas y asistentes *(Wizards)*: se han incluido dos nuevas herramientas y tres asistentes. *TeamSource* permite la creación de proyectos en equipo, compartiendo recursos y controlando las actividades de cada programador. *CodeGuard* es otra herramienta nueva para detectar errores en tiempo de ejecución que no se producen en tiempo de compilación. Entre los nuevos asistentes introducidos, hay uno para crear *applets* del *Panel de control* y otro para crear aplicaciones de tipo consola. También se ha incorporado un asistente para crear *DLLs* de forma más rápida y sencilla.

VERSIÓN 5.5 PARA LA LÍNEA DE COMANDOS

Además de la versión 5 de *Builder* con *IDE* para el desarrollo de programas, *Borland* sorprende agradablemente con una versión gratuita y totalmente funcional del compilador, algo que agradecerán todos los programadores: la *5.5*.

La diferencia fundamental de ésta es que, para compilar nuestros proyectos, tendremos que usar la línea de comandos. Puede resultar engorroso trabajar continuamente desde la línea de comandos, pero a cambio, se tiene un mayor control en las opciones de enlace, optimización, etcétera. *C++ Compiler 5.5* va provisto de soporte *ANSI/ISO C++*, *RTL*, *STL* y plantillas.

EL IDE (ENTORNO DE DESARROLLO)

Cuando ejecutamos *C++ Builder* aparece el entorno de desarrollo, comúnmente llamado *IDE*. En este entorno tenemos a nuestra disposición todas las herramientas para la creación de buenos programas.

El *IDE* de *C++ Builder* incluye un diseñador visual de formularios, *Inspector de objetos*, paleta de componentes, *Gestor de proyectos*, editor de código fuente, *debugger*, así como un herramienta para crear instalaciones.

Borland C++ Compiler 5.5 es un rápido compilador de 32 bits, gratuito y totalmente funcional

Podemos movernos libremente desde el *Editor de formularios* (donde insertaremos los controles) al *Inspector de objetos* para editar los parámetros iniciales de dichos objetos cuando se ejecute el programa. Si cambiamos propiedades relativas al código, como el nombre de un evento, el *Inspector de objetos* cambiará automáticamente el correspondiente código fuente. Además, los cambios en el código fuente, como el de nombre de un método, se reflejarán inmediatamente en el *Inspector de objetos*.

A partir de ahora, los archivos de formularios (*DFM*) se guardan por defecto con formato de texto, en lugar de binario. Pero si necesitamos utilizar formularios creados con esta versión 5 en versiones anteriores de *Builder*, necesi-

```
C:\Borland\bcc55\Bin>bcc32
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
Syntax is: BCC32 [ options ] file[s]     * = default; -x- = turn switch x off
  -2     * 80386 Instructions          -4      80486 Instructions
  -5       Pentium Instructions        -6      Pentium Pro Instructions
  -Ax      Disable extensions          -B      Compile via assembly
  -C       Allow nested comments       -Dxxx   Define macro
  -Exxx    Alternate Assembler name    -Hxxx   Use pre-compiled headers
  -Ixxx    Include files directory     -K      Default char is unsigned
  -Lxxx    Libraries directory         -M      Generate link map
  -N       Check stack overflow        -Ox     Optimizations
  -P       Force C++ compile           -R      Produce browser info
  -RT    * Generate RTTI               -S      Produce assembly output
  -Txxx    Set assembler option        -Uxxx   Undefine macro
  -Vx      Virtual table control       -x      Suppress autodep. output
  -aN      Align on N bytes            -b    * Treat enums as integers
  -C       Compile only                -d      Merge duplicate strings
  -exxx    Executable file name        -fxx    Floating point options
  -gN      Stop after N warnings       -iN     Max. identifier length
  -jN      Stop after N errors         -k    * Standard stack frame
  -lx      Set linker option           -nxxx   Output file directory
  -oxxx    Object file name            -p      Pascal calls
  -tWxxx   Create windows app          -u    * Underscores on externs
  -v       Source level debugging      -wxxx   Warning control
  -xxxx    Exception handling          -y      Produce line number info
  -zxxx    Set segment names
```

Figura 2.- Compilador gratuito de *Borland Versión 5.5* para la línea de comandos.

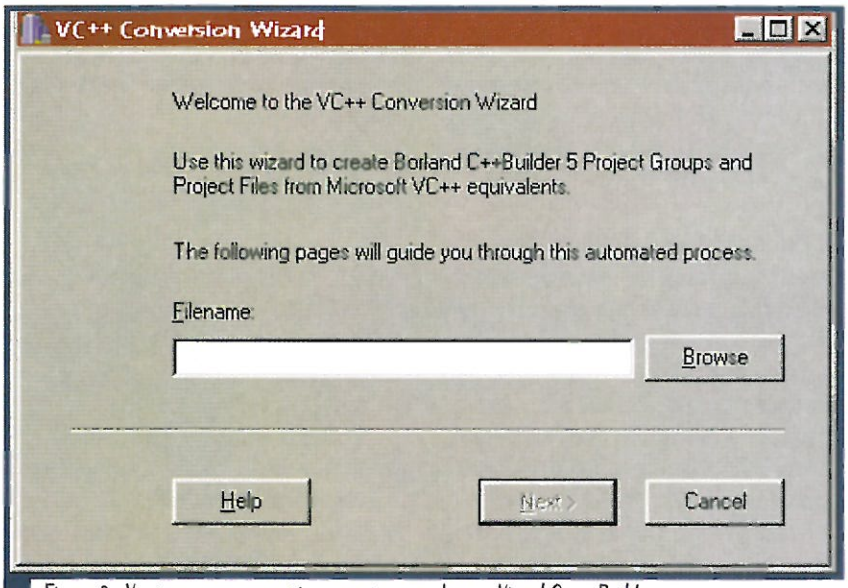


Figura 3.- Ventana para convertir proyectos creados en *Visual C++* a *Builder*.

taremos guardarlos explícitamente con formato binario. Para hacer esto, sólo tenemos que pulsar con el botón derecho del ratón sobre el formulario determinado, y desactivar la opción *Text DFM*.

Si queremos que todos los formularios creados se guarden con formato binario, sólo tenemos que ir al menú *Tools | Environment Options*.

El IDE incorpora una lista de tareas para saber qué queda por hacer en un proyecto

Cuando queramos importar nuestros nuevos formularios a versiones anteriores de *Builder* hay que prestar atención a un detalle: si utilizamos nuevas propiedades para los nuevos formularios (*THeaderSection.ImageIndex* y *THeaderControl.DragReorder*) devolverá errores al importarlos. Por este motivo, si queremos compatibilidad con versiones anteriores es conveniente no utilizar estas nuevas características.

Siguiendo con las novedades del *IDE* podemos definir varias plantillas de escritorio, darles un nombre y guardarlas de forma global. Podemos seleccionar la plantilla de escritorio que queramos, utilizando la opción de menú *View/Desktops*. También podemos especificar un escritorio para el *debugger*, para que se cargue durante la ejecución del programa que estemos creando.

Se han creado unas listas de tareas (*To-Do*) para mantener un recordatorio de las tareas que se deben completar en el proyecto. Así, podremos agregar notas en el código fuente o directamente, en la lista de tareas.

El *Inspector de Objetos* permite visualizar y filtrar las propiedades y eventos por categoría. Además, se ha creado un soporte para el *Inspector* con el que podamos ver imágenes (cursores, listas de imágenes, etcétera) en los cuadros combinados de listas.

Las preferencias del editor se han agrupado bajo un comando independiente (menú *Tools/Editor Options*). Ahora es más sencillo cambiar el editor a nuestro gusto. Además, el editor de código permite la inserción de líneas para dividir las funciones generadas en nuestro programa.

El menú *Tools/Environment Options/Preferences* permite seleccionar la creación automática de formularios. Para utilizar esta opción, debemos tener al menos un formulario creado de forma automática. Cuando la opción está desactivada, los formularios agregados al proyecto se colocan en la lista de formularios disponibles, en lugar de la de formularios creados automáticamente. Podemos cambiar la manera en que los formularios son listados mediante la opción de menú *Project/Options*, dentro de la pestaña *Forms*.

Podemos iniciar el *IDE* desde la línea de comandos utilizando diver-

TABLA 1. Entradas en b.BPF para la conversión.

1. *USEUNIT("archivo.CPP")*: una línea por cada archivo OBJ enlazado.
2. *USELIB("archivo.LIB")*: una línea por cada archivo LIB.
3. *USETLB("archivo.TLB")*: una línea por cada archivo TLB.
4. *USE("archivo.XXX")*: una línea por cada archivo misceláneo.
5. *USERC("archivo.RC")*: una línea por cada archivo RC.
6. *USERES("archivo.RES")*: una línea por cada archivo RES
7. *USEDEF("archivo.DEF")*: una línea por cada archivo DEF.
8. *#define WinMain*: para ejecutables.
9. *#define DllEntryPoint*: para DLL's.

TABLA 2. Parámetros de activación de CodeGuard.

OPCIÓN	DESCRIPCIÓN
Validate global and stack accesses	Crea plantillas y descriptores para los datos. Estos descriptores permiten que CodeGuard devuelva punteros inválidos a variables locales, globales y estáticas. Es recomendable tener esta opción siempre activada.
Validate the 'this' pointer on member function entry	Verifica el puntero this para la entrada a cualquier método. Esta opción reporta las llamadas a métodos de objetos eliminados o inválidos.
Validate pointer accesses	Genera llamadas para verificar los accesos de nuestros punteros. Con esta opción, el rendimiento de nuestro programa caerá a 5 o 10 veces más lento.
Casilla Default	Permite que las opciones seleccionadas se activen para cada nuevo proyecto que creemos.

sas opciones (muy útiles para el depurado de programas). Utilizando *switch /?* se inicia *C++ Builder* y visualiza la ayuda en línea para *switches* disponibles en la línea de comandos. El *switch /NP* hace que *C++ Builder* se inicie sin cargar ningún proyecto.

Los componentes *NetMasters* ahora aparecen en una página distinta (*FastNet*) de la paleta de componentes.

La compatibilidad entre Builder y Visual C++ es excelente

El componente *TCppWebBrowser* de la página *Internet* que importa el *ActiveX Internet Explorer*, sustituye al componente *THTML*.

Para finalizar con el *IDE*, mencionaremos la nueva herramienta visual que facilita la creación y el mantenimiento de módulos de datos.

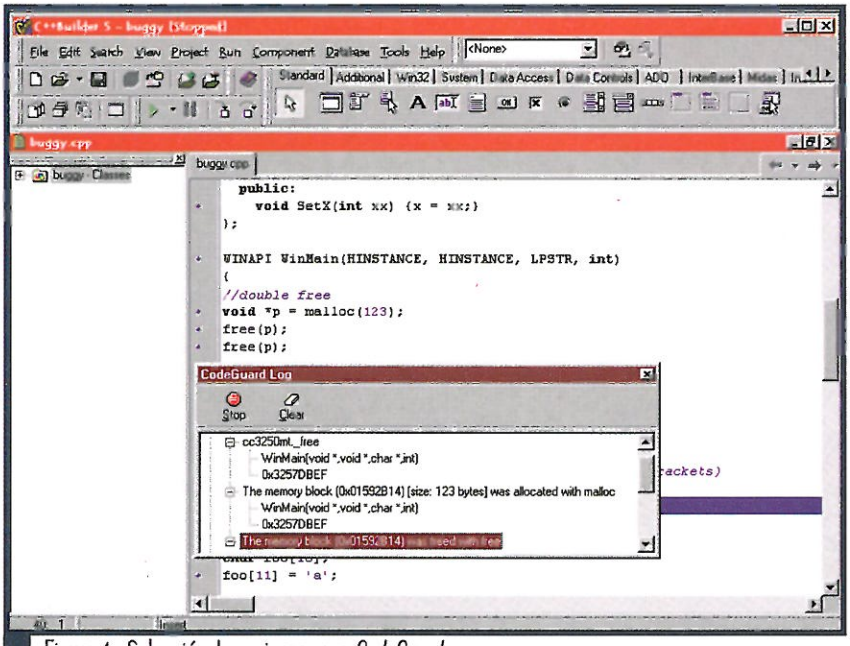


Figura 4.- Selección de opciones para *CodeGuard*.

CONVERSIÓN DE PROYECTOS CREADOS CON VISUAL C++

Builder puede convertir proyectos y espacios de trabajo (*Workspaces*) creados con *Visual C++ 5* y *6*. Para ello, debemos seleccionar la opción del menú *Tools/Visual C++ ProjectConversion Utility*. Las extensiones de estos archivos son *dsp* y *dsw*.

No se pueden realizar conversiones sobre archivos que tengan el atributo de sólo lectura activado, como aquellos que se encuentran almacenados en *CD-ROM*.

LISTADO 1. Programa de compilación incluido en *Builder*.

```
#include <vcl.h>
#pragma hdrstop
USERES("buggy.res");

class MyClass {
  private:
    int x;
  public:
    void SetX(int xx) {x = xx;}
};

WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
//Se utiliza free dos veces, cuando el puntero ya se ha //liberado utilizando el primer free.
void *p = malloc(123);
free(p);
free(p);

//Acceso al puntero después de haber sido liberado
int *i = (int*)malloc(sizeof(int));
free(i);
*i = 99;

//se reserva memoria con new[] y se libera con delete sin //utilizar los corchetes
char *c = new char[64];
delete c;

//foo apunta al elemento 11 cuando solamente hay 10 //elementos en el array
char foo[10];
foo[11] = 'a';

//copia una cadena en otra que su tamaño tiene un carácter //menos
char *name = "Chris";
char *myname = new char[strlen(name)];
strcpy(myname, name);
delete[] myname;

//llama a la función miembro de la clase después de que la //instancia de la clase ha sido liberada
MyClass *instance = new MyClass;
delete instance;
instance->SetX(10);

//Se reserva memoria para un puntero que posteriormente no //se libera con delete
TStringList *sl = new TStringList;
sl->Add("Hello there");
return 0;
}
```

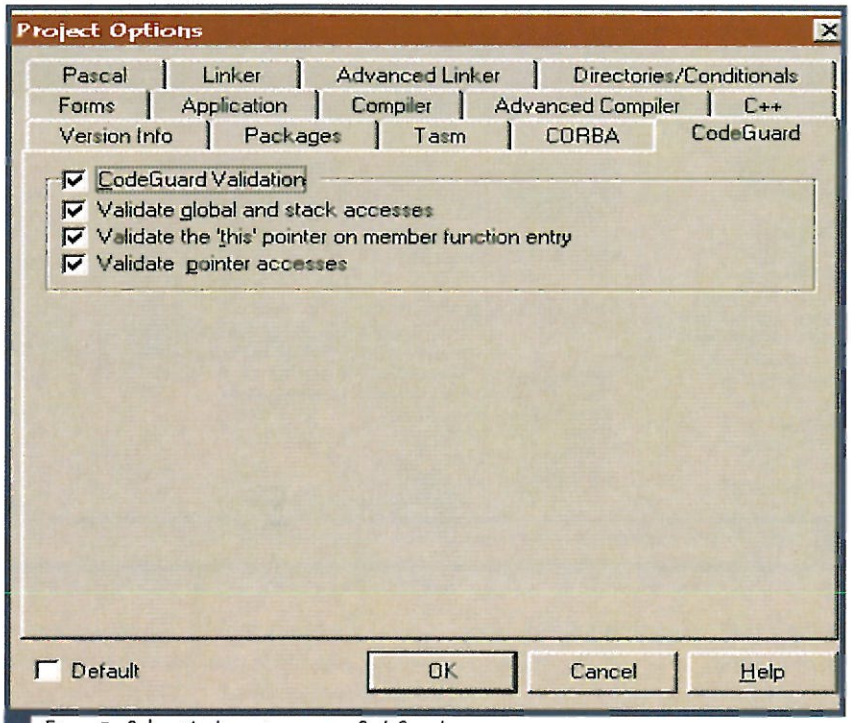


Figura 5.- Selección de opciones para *CodeGuard*.

A continuación se relacionan los tipos de proyectos creados con Visual C++ que podemos convertir:

- Aplicaciones *ATL COM*.
- Controles *MFC ActiveX*.
- Aplicaciones *MFC (DLL)*.
- Aplicaciones *MFC (EXE)*.
- Aplicaciones *Win32*.
- Aplicaciones para consola *Win32*.
- *Win32 DLL*.
- *Win32 Static Library*.

Al hacer la conversión, si se especifica un archivo de tipo "espacio de trabajo" (*workspace*), *Builder* creará un archivo *.BPG*.

Si se especifica un fichero de proyecto, *Builder* creará un archivo *.BPR* y otro *.BPF* con el mismo nombre que el archivo de proyecto.

En la Tabla 1 podemos ver las entradas que se agregan al archivo *.BPF* cuando se realiza una conversión.

Al realizar conversiones de proyectos creados con *MFC's* es conveniente seleccionar el modo del proyecto (*Debug* o *Release*).

Si encontramos errores al realizar la conversión de un proyecto necesitaremos hacer algunos cambios (mínimos) en el código fuente.

CODEGUARD

CodeGuard es una nueva herramienta de *Builder* que permite depurar programas en tiempo de ejecución. Solo está disponible en las versiones *Professional* y *Enterprise*. Vamos a detenernos más en esta herramienta, ya que ofrece muchas posibilidades para el programador.

Con *CodeGuard* podemos detectar errores que en tiempo de compilación no violan la sintaxis del código, pero que en tiempo de ejecución provocan la caída del programa. Un ejemplo claro lo vemos cuando intentamos acceder a posiciones de memoria o recursos que ya han sido liberados, y por lo tanto, no podemos dirigir. *CodeGuard* chequea la falta de memoria, fallos en la localización de memoria, etcétera.

Puesto que C/C++ es un lenguaje en el que se utilizan continuamente punteros y por lo tanto, direcciones de memoria, esta herramienta será de valiosa utilidad para cualquier programador.

CodeGuard permite detectar errores runtime que no aparecen en tiempo de compilación

CodeGuard tiene dos tipos de protección:

- Utilización de la memoria y los recursos: comprueba la falta de memoria, su reserva y liberación, ficheros no válidos y *handles*. También verifica punteros y operaciones con éstos. *CodeGuard* emite mensajes de error si por ejemplo, intentamos acceder a memoria o a recursos que previamente han sido liberados.

- Validación de llamadas a funciones: *CodeGuard* verifica los argumentos de las funciones y devuelve el fallo con el valor de retorno de la propia función.

CONFIGURACIÓN DE CODEGUARD

CodeGuard sólo detectará errores en programas ejecutables compilados con las opciones del compilador para *CodeGuard* activadas.

CodeGuard puede utilizarse con la librería estática (*CG32.LIB*) o la dinámica (*CG32.DLL*). En estas librerías hay funciones que verifican

las funciones de nuestro programa y los valores de retorno de las mismas.

Si queremos utilizar la protección de CodeGuard en nuestros programas, debemos acceder al menú *Project/Options,* y aquí, seleccionar la pestaña *CodeGuard.* Dentro de esta opción, podremos seleccionar el nivel de protección que queremos obtener.

Al seleccionar cualquiera de las opciones, es posible que notemos una caída en el rendimiento de nuestro programa, pero esto no debe extrañar, si se tiene en cuenta que *CodeGuard* debe seguir paso a paso toda la gestión de recursos que hace nuestro programa.

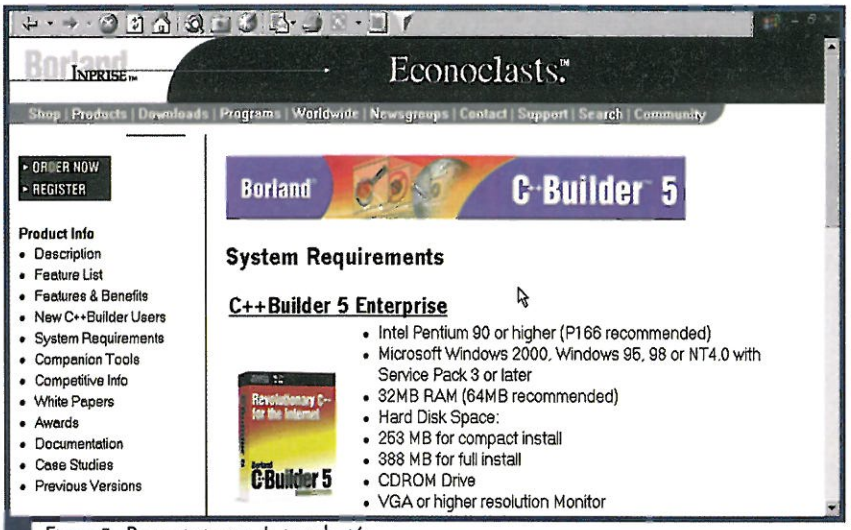


Figura 7.- Requerimientos de instalación.

C++ Builder 5 es una de las mejores herramientas de desarrollo en C y C++

Bebemos recordar que cada vez que cambiemos alguna de las opciones de *CodeGuard,* tendremos que reconstruir completamente nuestro proyecto.

En la Tabla 2 podemos ver los parámetros que podemos activar para *CodeGuard.*

En el Listado 1 podemos ver un pequeño programa (incluido en *Builder*) que compila perfectamente, pero que en tiempo de ejecución provoca errores en la gestión de acceso a memoria de los punteros.

En la figura 4 podemos ver el resultado de *CodeGuard* al ejecutar este programa.

Si comprobamos detenidamente el listado podremos observar que continuamente se están gestionando mal las localizaciones de memoria y los accesos de los punteros, aunque la sintaxis sea correcta y el programa pueda compilarse perfectamente.

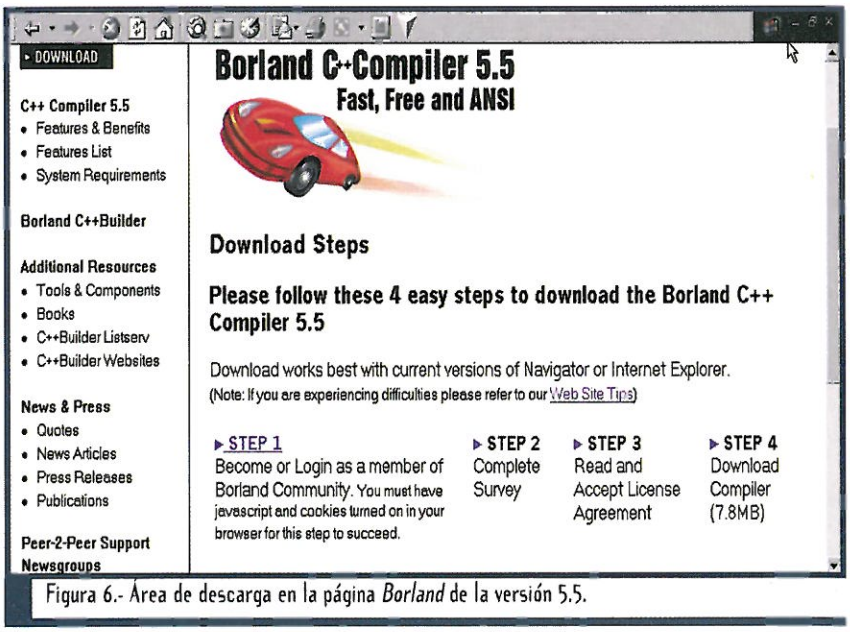


Figura 6.- Área de descarga en la página *Borland* de la versión 5.5.

CONCLUSIÓN

Definitivamente, esta última versión va a subir el rasero a sus competidores en el mercado. Tras las primera pruebas, podemos afirmar que la evaluación inicial de *C++ Builder 5* es francamente satisfactoria. A pesar de los errores que podamos encontrar, como queda dicho más arriba, creo que nos encontramos ante una de las mejores herramientas puestas al alcance de los programadores de *C* y *C++*.

De hecho, la mayoría de los *bugs* y errores de programación vienen dados por la incorrecta utilización de punteros y accesos a memoria en tiempo de ejecución.

DA RIENDA SUELTA A TU CREATIVIDAD E IMAGINACIÓN CON:

MASTER COREL DRAW 9

HASTA UN **55%** DE DESCUENTO

Suscríbase ahora al Master Corel Draw, y junto a la colección de 50 fascículos y 50 CD-ROM recibirá la *Suite completa de Corel Draw 9.0. versión estudiante.*

Más de 1.000 páginas de teoría y práctica llenos de los más útiles consejos para convertirse en un auténtico Master del diseño gráfico.

Más de 200 programas y utilidades del diseño gráfico, más de 500 fotografías profesionales y decenas de Webs de diseño para navegación off-line.

Ejercicios basados en las explicaciones del curso que le ayudarán a perfeccionar tu técnica.

Diploma acreditativo de haber recibido y aprovechado el *Master Corel Draw* de diseño gráfico.

Con 5 carpetas para almacenar sus fascículos y un archivador para ordenar los CD-Rom.

Desarrolle toda la capacidad de las nuevas herramientas que le ofrece Corel Draw 9: Ilustraciones vectoriales, Corel Photo-Paint 9 (edición de imágenes y bitmaps), C. Texture (Texturas realistas y naturales), C. Capture (Aplicación para la captura de pantallas), 25.000 imágenes clipart de alta calidad (10.000 nuevas para C. Draw 9), y muchas otros extras.

BOLETÍN DE SUSCRIPCIÓN

OFERTA VÁLIDA SÓLO PARA ESPAÑA

SI, deseo suscribirme al **MASTER COREL DRAW 9**

PRECIO EN EL QUIOSKO

795 Ptas. cada entrega de 1 fascículo y un CD-ROM
Total 50 fascículos, 50 CD-ROM y 5 tapas: 950 ptas.
1 carpeta para guardar los CD-ROM.
TOTAL DE LA COLECCIÓN COMPLETA: 44.955 Ptas.

CONDICIONES DE PAGO

(MARQUE LA OPCIÓN DESEADA)

- ❑ Pago en 12 mensualidades de **2.995** Ptas.: **35.960** Ptas. Ahorro del **20%**.
- ❑ Pago en 6 mensualidades de **5.245** Ptas.: **31.470** Ptas. Ahorro del **25%**.
- ❑ Pago en 3 mensualidades regalo del Corel 9. En la última mensualidad ahorro del **45%**. Cada mensualidad de **26.485** Ptas.: **79.455** Ptas.
- ❑ Pago al contado con el Corel 9 de regalo y un **55%** de descuento: **38.212** Ptas.

UTILICE MAYÚSCULAS PARA RELLENAR ESTA TARJETA

Nombre y apellidos ..... F. nacimiento.....
Domicilio ..... C.P. .....
Ciudad ..... Provincia ..... Telf. .....
e-mail ..... Profesión .....

FORMAS DE PAGO:

❑ Con **cargo a mi tarjeta VISA** Nº ____/____ /____ /____ Caduca .....

❑ **Domiciliación bancaria**

Ruego a UD. que se sirva cargar en mi ❑ cuenta corriente ❑ libreta de ahorro el/los recibos que le serán presentados por REVISTAS PROFESIONALES, S.L. como pago de mi suscripción al **MASTER COREL DRAW 9 DE DISEÑO GRÁFICO**

CÓDIGO CUENTA CORRIENTE

ENTIDAD	OFICINA	DC	Nº CUENTA

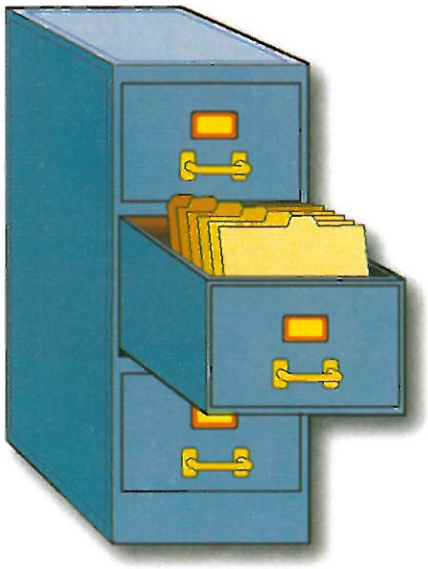
❑ **Cheque** a nombre de REVISTAS PROFESIONALES, SL.

❑ **Contra reembolso** del importe más gastos de envío.

❑ **Giro Postal** (adjunto fotocopia del resguardo)*.

Firma

*Esta forma de pago es exclusivamente para pagos al contado



Aplicaciones de bases de datos con Delphi 5 (V)

Juan Luis Ceada Ramos.
Programador en ARCABE Formación y Servicios Informáticos

En este número modificaremos la aplicación desarrollada, para que pueda funcionar con diferentes gestores de bases de datos.

ACCEDER A LOS DATOS

Como indicamos en la pasada entrega, lo primero que vamos a ver es cómo podemos conectar nuestras aplicaciones *Delphi* con prácticamente cualquier sistema gestor de bases de datos del mercado. En este caso nos centraremos en *Interbase* y *Oracle*.

También veremos cómo conectar con bases de datos de escritorio, como *Paradox* y *Access*. Para trabajar con las primeras, es necesario que en nuestra máquina (o en alguna a la que tengamos acceso vía red) se encuentre activo el servidor *Interbase/Oracle*, en función del servidor utilizado.

ACCESO NATIVO

Delphi incluye sus propios mecanismos para acceder a un gran número de gestores de bases de datos. La lista incluye *DB2*, *Access*, *Paradox*, *Dbase*, *Informix*, *Oracle*, *SQL Server*, etcétera. El tamaño de esta lista se incrementa con cada nueva versión de *Delphi*.

El acceso nativo es la forma recomendada para acceder a los datos. Sólo en caso de no contar con el *driver* apropiado deberíamos recurrir a *ODBC*. El acceso nativo es más rápido, y permite trabajar con los datos de una forma más eficiente. El problema de este método es que si actualizamos nuestro sistema de bases de datos con una nueva versión, y ésta incluye novedades que hacen que nuestro

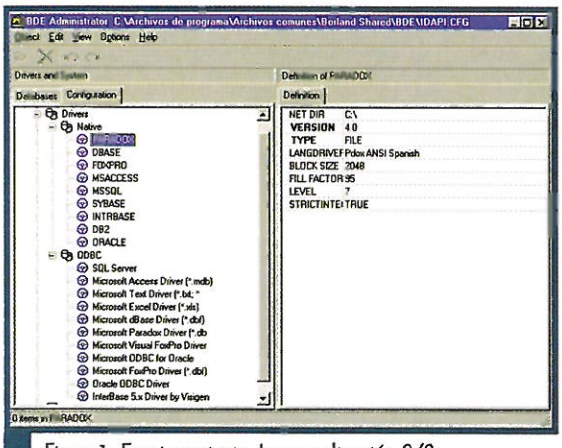
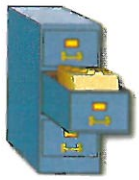


Figura 1.- Funcionamiento de una aplicación C/S.



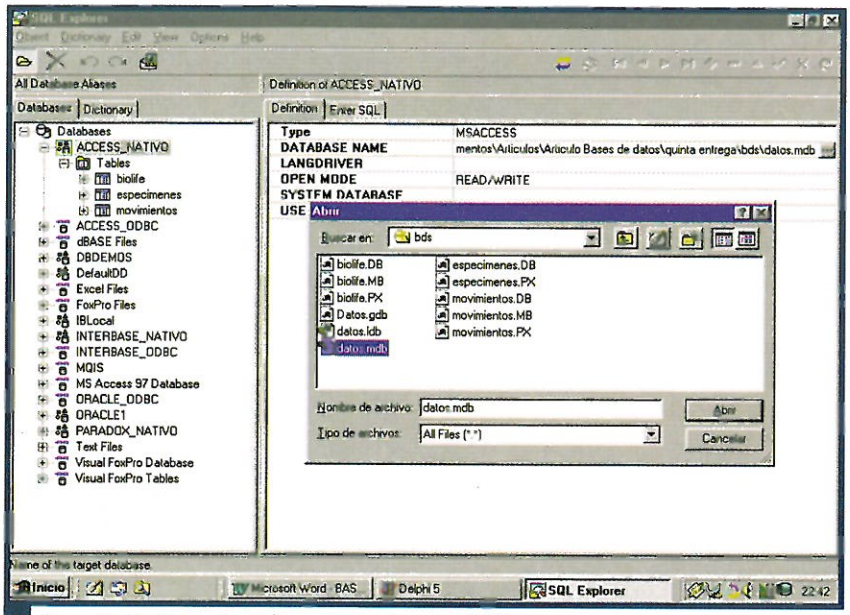


Figura 2.- Configurando un alias para acceder a una BBDD *access*.

driver no funcione en la misma, nos veremos obligados a utilizar un nuevo controlador (también llamado *SQL Link* si trabajamos con *SGBD SQL*). Si éste no se encuentra disponible, evidentemente nuestra aplicación no podrá acceder a los datos.

Delphi proporciona acceso nativo a los principales gestores de bases de datos

De todas formas, con cada versión nueva de *Delphi/BDE*, se actualizan los *SQL Links*, añadiendo compatibilidad con las nuevas versiones de *SGBD's* que aparecen en el mercado, por lo que la situación anterior es poco probable.

Los *SQL Links* a los que antes nos hemos referido, no son más que un conjunto de controladores manejados por el *BDE* que permiten conectar con servidores de bases de datos. Mediante la creación de sentencias *SQL*, los *SQL Links* permiten trabajar con este tipo de bases de datos usando los mismos métodos de navegación que usaríamos con bases de datos de escritorio (*First*, *Last*, *Edit*, *Post*, etcétera.)

PARADOX Y ACCESS

En ambos casos, el acceso nativo está disponible en todas las versiones de *Paradox* hasta la versión 97 de *Access*. Al tratarse de bases de datos de escritorio, no es necesario que exista un servidor de bases de datos en ejecución. Por tanto, podemos trabajar con bases de datos *Access*, por ejemplo, sin que en nuestra máquina esté instalada dicha aplicación.

Lo único que tendremos que hacer, es crear un nuevo alias que apunte a los ficheros de datos. En el caso de *Paradox*, más que apuntar a un fichero en concreto, tendremos que hacer que el alias apunte a un directorio, puesto que en este sistema, cada tabla se almacena como un fichero independiente. El alias debe crearse como tipo *STANDARD*.

En el caso de *Access*, sí que tendremos que hacer que el alias apunte al fichero *MDB* que contenga las tablas (en nuestro caso Datos.mdb). Para crear los alias podemos usar el *BDE Administrator*, o bien *SQL Explorer*. El alias debe crearse como de tipo *MSACCESS* (durante la instalación de *Delphi*, se nos dejó indicar si el *driver MSACCESS* permitiría acceder a ficheros *Access 97* o *Access 95*).

Es posible modificar la versión de *Access* a la que accederemos desde el *BDE Administrator*. Basta con realizar la siguiente secuencia de operaciones:

```
Configuration -> Drivers ->
    Native -> MSACCESS -> DLL32
```

y elegimos la *DLL* más adecuada (*IDDAO32.DLL* para *Access 95* y *IDDA3532.DLL* para *Access 97*). Es de suponer que la próxima revisión del *BDE* proporcionará acceso nativo a *Access 2000*.

En ninguno de los dos casos (*Access* o *Paradox*) tendremos que cambiar ningún parámetro del *alias*. Normalmente, con los que vienen por defecto será suficiente. Una vez creados, podemos ver si nuestra operación ha tenido éxito: haciendo doble clic sobre el alias, debería desplegarse la lista de tablas a las que tenemos acceso.

En el caso particular de *Access*, se nos pedirá que introduzcamos un *login* y *password* para poder acceder a los datos. Deberemos pulsar *OK* (a menos que la base de datos esté protegida mediante *password*, en cuyo caso deberemos teclearla antes de pulsar *OK*).

INTERBASE Y ORACLE

Interbase es un sistema gestor de bases de datos potente, barato, sencillo de configurar y bastante rápido, destinado a aplicaciones de tamaño mediano. Además, parece

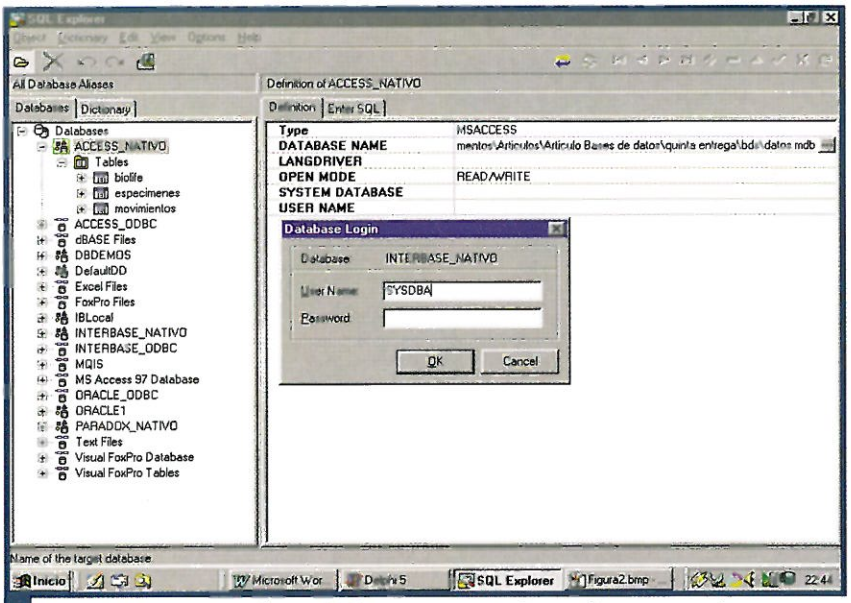


Figura 3.- Para acceder a algunas *BBDD* es necesario introducir un *login* o *password*.

que sus creadores se han apuntado a la moda del *software* libre, y hay rumores de que la versión 6.0 será gratuita.

Como gestor de bases de datos no tiene nada que envidiar a otros más potentes, tales como *Oracle*. Además, como es el *SGBD* que se distribuye con *Delphi*, se supone que configurar el acceso a bases de datos *Interbase* es relativamente sencillo. Lo único que tendremos que hacer es crear un nuevo alias de tipo *INTRBASE*. De todos los parámetros del alias, basta con que modifiquemos *SERVER NAME*. Aquí indicaremos la ubicación del fichero *GDB* que contiene la base de datos *(Datos.gdb)*.

Interbase es el SGBD incluido en Delphi 5 Enterprise

Al intentar conectar con la base de datos, se nos pedirá el nombre de usuario y *password*. Por defecto, el administrador de un sistema *Interbase* es *SYSDBA* y su *password* es *masterkey*.

En *Oracle* el proceso resulta algo más complicado. Aquí, las bases de datos no están en un fichero al que podamos apuntar. Es necesario que usando la herramienta *SQL Net8 Assistant*, creemos una conexión a la base de datos que deseemos. Después, crearemos un alias de tipo *ORACLE*.

El parámetro más interesante del alias es *SERVER NAME*, donde deberemos indicar el nombre de la conexión que habíamos creado previamente desde *Oracle*. Al igual que en *Interbase*, tendremos que proporcionar el nombre de usuario y una contraseña, que para nuestras pruebas será *Scott / tiger*. Dicho usuario es creado por *Oracle* durante la instalación y será en su espacio de usuario donde creemos las tablas.

ACCESO ODBC

ODBC es un estándar, creado por *Microsoft*, que permite acceder a cualquier base de datos, que proporcione un controlador *ODBC* desde nuestras aplicaciones *Delphi*.

Toda su configuración se realiza desde el *ODBC Data Sources*, situado en el *Panel de control*. En la pestaña *drivers*, podremos ver a qué bases de datos podremos acceder mediante *ODBC*. Su número depende de las aplicaciones que haya instaladas en nuestro equipo. Evidentemente, si no tenemos un *driver ODBC* de *Oracle*, no podremos acceder a bases de datos *Oracle* desde ninguna aplicación usando *ODBC*.

Sólo podemos usar ODBC si tenemos los drivers adecuados

En la pestaña *User DSN*, tenemos las conexiones definidas por el usuario. Si disponemos de un controlador *ODBC* para *Access*, es en esta pestaña donde tendremos que configurar el acceso a cada una de las bases de datos. En este caso habría una conexión por cada fichero *MDB* al que nos queramos conectar.

La configuración de un acceso a una *BD* no es estándar, ya que cada *driver OBDC* proporciona su propio diálogo de configuración. En todos los casos habrá que rellenar al menos, el nombre que recibirá la conexión, que es el que usan las aplicaciones para acceder a los datos. Además, habrá que indicar a qué datos queremos acceder. Dependiendo del *driver*, el cuadro de diálogo tendrá más o menos opciones.

ACCESS

En este caso, bastara con abrir el *ODBC Data Sources*, y en la pestaña *User DSN*, pulsar el botón *Add*. Aparecerá un cuadro de diálogo permitiéndonos elegir el *driver ODBC* que vamos a usar. Elegiremos *Microsoft Access Driver*, apareciendo entonces un cua-

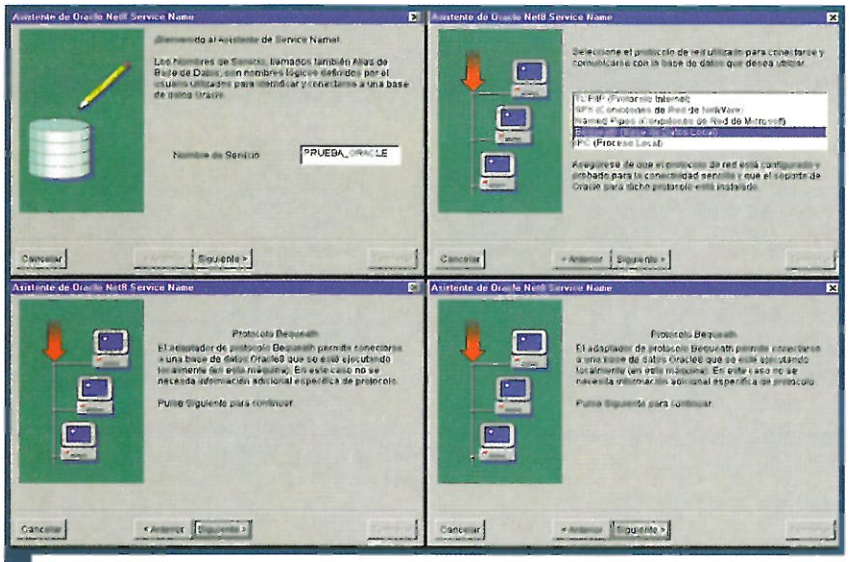


Figura 4.- Creando una conexión con *Net8 Assistant*.

dro de diálogo similar al de la Figura 6. Daremos un nombre a la fuente de datos, una descripción y seleccionaremos la base de datos deseada (mediante el botón *select* del fichero *Datos.mdb).*

Con esto ya tenemos configurado nuestro acceso *ODBC*. Si abrimos *SQL Explorer*, veremos cómo aparece la conexión que acabamos de crear en la zona reservada a los alias. Con un doble clic abriremos dicha conexión y podremos acceder a las tablas de la base de datos.

INTERBASE

El proceso es similar al anterior, ya que basta con crear una conexión de usuario de tipo *INTERBASE*. Después, rellenaremos las opciones *Data Source Name* y seleccionaremos el fichero *Datos .gdb*. Con esto, ya tendremos el enlace *ODBC* creado. En la máquina de pruebas, es posible introducir un usuario por defecto para acceder a la base de datos, aunque esta opción es posible que no esté presente en otros equipos, puesto que como hemos dicho antes, los cuadros de diálogo dependen de la versión del *driver ODBC* que poseamos.

ORACLE

El caso de *Oracle* es igual de sencillo. En la Figura 7 se muestra el cuadro de diálogo que instala el *driver ODBC* de *Oracle 8.0.4*. Como se puede apreciar, además de darle un nombre al *Data Source* tendremos que indicar con qué conexión (creada con *SQL Net8 Assistant*) y con qué usuario accederemos a las bases de datos.

LAS PRUEBAS

La aplicación del ejemplo que vamos a desarrollar se basa en la que ya desarrollamos en la tercera entrega. En realidad, es la misma, aunque con pequeños cambios.

CAMBIOS EN LAS TABLAS

Necesitamos que nuestra aplicación funcione con diferentes formatos de base de datos. Esto va a suponer que tengamos que realizar algunos cambios en las tablas. Evidentemente, hay que adaptarse al sistema más restrictivo. La aplicación original, usaba tres tablas *Paradox: biolife.db, especimenes.db y movimientos.db.*

Para asegurarnos de que no haya problemas al pasar las tablas a otros formatos tendremos que:

- Renombrar todos los campos que contengan caracteres extraños (como espacios, vocales acentuadas, etcétera.)
- Cambiar todos los campos autoincrementales por campos flotantes.
- Reducir el número de campos *memo* o de gráficos, a uno por tabla.

En la práctica, las aplicaciones se diseñan para un único sistema de BBDD

Sobre el punto 2, hay que añadir que *Interbase* no soporta campos auto-incrementales. Habría que obtenerlos mediante un generador. Por tanto, es mejor convertir ese tipo de campos a flotante y simular el auto-incremento desde *Delphi*. ¿Por qué a flotantes y no a enteros largos? La razón es que en *Oracle* los campos numéricos son siempre flotantes. Insistimos en que hay que adaptar las tablas al sistema más restrictivo.

Sobre el punto 3 conviene decir que *Oracle* no permite más de un campo *Long* (que es a lo que se convierten los *memos* o *graphics*) por tabla.

En el directorio *BBDD* (en el *CD-ROM* de la revista), se encuentran las tres tablas (en formato *Paradox*) ya revisadas. Está claro que no estamos aprovechando las características avanzadas de cada sistema, pero recuerde: esto es sólo un ejemplo. En la práctica, nadie programa una aplicación para que soporte múltiples sistemas de bases de datos.

TRASPASAR DATOS

Para pasar los datos de un formato a otro, hay que dar tres pasos:

- Crear bases de datos vacías en cada uno de los formatos. Por ejemplo, en *Access* crearemos un fichero *Datos.mdb*, pero sin crear ninguna tabla. En *Oracle* no es necesario crear la base de datos, usaremos la que trae de ejemplo, identificándonos como usuario *Scott*.

- Crear los alias que apuntan a las bases de datos como hemos explicado con anterioridad. Hay que crear alias para acceso nativo y mediante *ODBC*. Para nuestras pruebas, tendremos que crear los siguientes alias: *paradox_nativo, access_nativo, access_odbc, interbase_nativo, interbase_odbc, oracle_nativo, oracle_odbc*.

- Traspasar los datos de *Paradox* al resto de los sistemas. Para ello usaremos la utilidad *DataPump* (proporcionada con *Delphi*). Esta utilidad es muy sencilla de usar, y basta con indicar el alias de origen de datos (*PARADOX_NATIVO*), el alias de destino (cualquiera de los otros), seleccionar las tablas que traspasar...y poco más.

Con *Oracle*, *Datapump* tiene una extraña tendencia a transformar un campo flotante en un *NUMBER(1,0)*, es decir, un flotante de un solo dígito. En ese ca-so, habrá que modificar el campo a *NUMBER(12,0)*.

En el *CD-ROM* se han incluido las bases de datos *Access*, *Interbase* y *Paradox*, convenientemente modificadas, y con el traspaso de datos realizado. El caso de *Oracle* es especial, ya que al usar la base de datos de ejemplo, no es posible su distribución en el *CD*. Así que los lectores interesados en hacer pruebas con *Oracle*, tendrán que realizar el tercer paso por su cuenta. Se deben copiar todos los ficheros en el disco duro, y quitar la protección contra escritura antes de crear los alias y hacer el traspaso de datos.

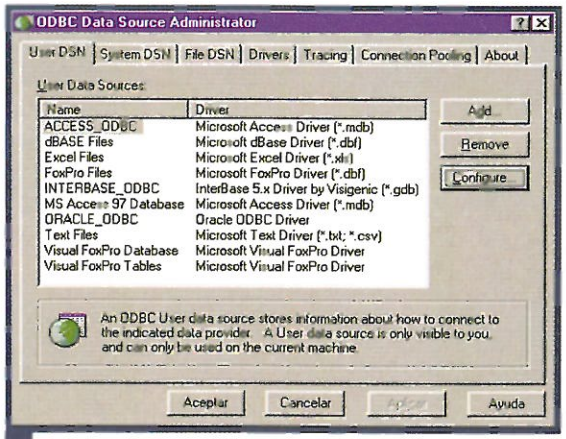


Figura 5.- *ODBC Data Sources* en el panel de control.

CAMBIOS EN EL DATAMODULE

En primer lugar, se ha añadido un componente *TDatabase*. Por defecto, en su propiedad *Alias Name*, seleccionamos el alias *PARADOX_NATIVO*. Además, creamos un alias local (propiedad *DataBaseName* igual a *MiAlias Local*). El resto de las propiedades quedan sin modificar.

El componente TDatabase permite gestionar las transacciones

La propiedad *DataSource* de los *TTables* y *TQuerys* apunta ahora al alias local, es decir, *MiAliasLocal*. Por cada uno de los *TTables* tendremos que abrir el *Editor de Campos* (haciendo doble clic en el *TTable*), eliminar los campos que hubiese y añadirlos de nuevo. Con esto conseguiremos que se creen los componentes de acceso a campos, adaptados a los cambios introducidos en las tablas con respecto a la aplicación de la tercera entrega.

También se ha añadido un *TStoredProc* que permitirá lanzar el procedimiento almacenado *BorrarHijos* (ver código del evento *BeforeDelete* de *EspecimenTbl*). Dicho procedimiento sólo se usará en caso de que trabajemos con *Interbase* (ya que al estar la *BBDD* en un único fichero *GDB*, es posible distribuir también el procedimiento almacenado). El código de dicho procedimiento se muestra en la Figura 10. Como su nombre indica, elimina todos los movimientos de un espécimen. Como parámetro (propiedad *Params*) hemos añadido *IDEspecimen*, de tipo *Float*. Este parámetro indica al procedimiento qué movimientos debe borrar.

La verdad, es que este tipo de operaciones suelen incluirse en *triggers*, que saltan automáticamente al borrar un espécimen, pero me pareció un ejemplo sencillo para aprender a usar procedimientos almacenados. Sólo tendremos que hacer:

```
DatabaseName = MiAliasLocal
StoredProcName = BorrarHijos.
```

Además, para simular los campos auto-incrementales (los campos *ID*

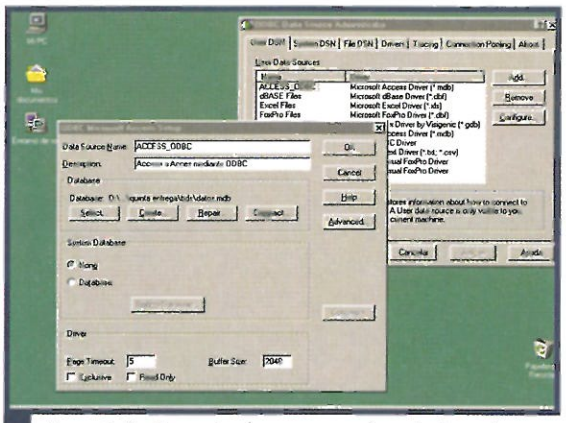


Figura 6.- Configurando el acceso a una base de datos *Access*.

de las tablas *Especímenes* y *Movimientos*), tendremos que añadir código a los eventos *AfterInsert* de los *TTable*. Dicho código se limita a obtener el ultimo *ID* de la base de datos e incrementarlo en uno (Figura 11).

Por ultimo, no hay que olvidar que las *TTables* usan por defecto tablas *Paradox*, por lo tanto es probable que en la propiedad *Table Name* aparezca el nombre del fichero con extensión. Pues bien, la extensión hay que eliminarla, para evitar errores al abrir las tablas en otro formato.

CAMBIOS EN UPRINCIPAL.PAS

Esta unidad almacena el formulario principal de la aplicación. En el evento *OnActivate,* lanzamos el diálogo de selección de alias. Este consta de un *combo* al que hemos añadido los posibles alias a los que nos vamos a conectar. Basta con elegir uno de ellos, y pulsar *OK*. Si todo es correcto (existe el alias, existe la base de datos, y en caso de *Interbase/Oracle*, el servidor está activo), se abrirán las tablas correspondientes.

Es posible que se nos solicite *login/password* de acceso. Esto se puede evitar insertando estos parámetros en la propiedad *Params* del *TDataBase y haciendo:*

```
TDatabase.LoginPrompt = False
```

Como se puede apreciar en el código, para cambiar de base de datos, sólo tendremos que asignar el nuevo alias al *TDatabase* y volverlo a conectar.

También hemos introducido en esta versión de la aplicación el uso de transacciones. Las transacciones empiezan cuando se pulsa el botón de insertar o editar (*MiDatabase.StartTransaction*). Terminan al aceptar los cambios (*MiDataBase.Commit*), o al cancelarlos (*MiDataBase.RollBack*).

TStoredProc permite ejecutar procedimientos almacenados

Como dijimos en la pasada entrega, un *commit* almacenará en las tablas todos los cambios que hicimos desde que empezamos a editar los datos. Un *rollback* cancelará cualquier actualización en la base de datos y la dejará en el estado en el que se encontraba antes de comenzar la transacción. Después de ejecutar estos métodos, es conveniente refrescar las tablas.

En la aplicación de la tercera entrega, si dábamos de alta un espécimen, luego efectuábamos varios movimientos, y por ultimo cancelábamos la operación, sólo se cancelaba la actualización de los registros que estuviesen en modo de edición. Es decir, a lo sumo el registro espécimen, y el último movimiento introducido. Esto provocaba que en la base de datos quedasen movimientos "huérfanos". La solución es simple: borrar todos los movimientos antes de cancelar, o más fácil aún, hacer un *rollback* de la transacción.

CAMBIOS MENORES

En general, habrá que actualizar los *TDBEdits* y similares de todos los formularios, para que se adapten a los nuevos cambios. Por ejemplo, en el formulario Especies, habrá que eliminar el control *DBImage*, puesto que hemos eliminado las imágenes de las tablas. Otro ejemplo: el *TDBEdit Localización* del formulario *FGrid* debe apuntar al campo *Localizacion*, y no Localización (nótese que este último tiene acento).

FINALMENTE

Sólo tendremos que ejecutar la aplicación y elegir el alias con el que queremos trabajar. Funcionará perfectamente, con independencia de la base de datos a la que nos conectemos. Si deseamos cambiar de base de datos, habrá que cerrar y volver a abrir la aplicación. En el próximo número os presentaremos el sexto y último artículo de esta serie.

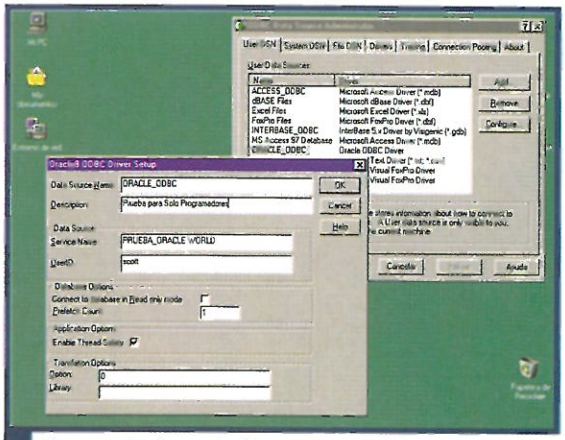
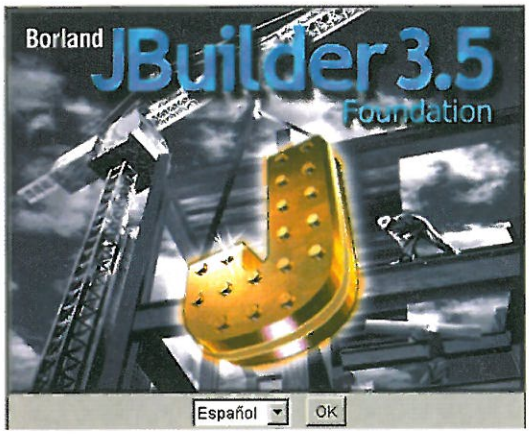


Figura 7.- Configurando el acceso a una *BBDD ORACLE*.



Entorno RAD JBuilder 3.5

Javier Sanz Alamillo.

Ya está disponible *JBuilder 3.5*, el entorno de desarrollo rápido para *Java*, apto para diferentes plataformas y destinado a mejorar y ampliar las posibilidades de su precedente, la versión 3.

INTRODUCCIÓN

El mercado de los entornos rápidos de desarrollo para *Java* está experimentando constantes cambios, traducidos en la imparable aparición de novedades y mejoras. *Borland* contribuye a esa evolución con la presentación de un nuevo y completo entorno multiplataforma, por lo que se puede afirmar que actualmente es el mejor elaborador de entornos de desarrollo *Java*, por la calidad de sus productos (¡qué desarrollador no conoce algún producto *Borland!*), por el esfuerzo realizado para que éstos sean reconocidos, y por ofrecer el primer entorno para *Java* multiplataforma, dejando así a sus competidores más directos algo rezagados. Estos últimos basan sus productos principalmente en el desarrollo de una única plataforma -*Windows*-, lo que les hace estar fuera de ciertos mercados cada vez más importantes, como son el mundo *Linux* y el *Unix* tradicional.

A continuación, se presenta *JBuilder 3.5*, la punta tecnológica para el desarrollo en *Java* de *Borland*, que hará las delicias de los seguidores de los productos de este gran constructor de *software*.

Figura 1.- *Web JBuilder 3.5*

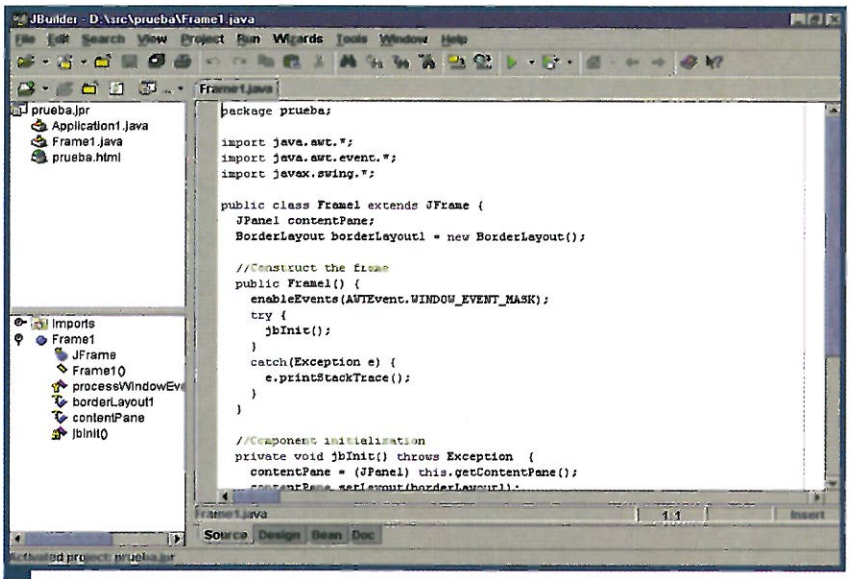


Figura 2.- Entorno de desarrollo de *JBuilder 3.5*.

JBUILDER 3.5

Sin apenas tiempo para sacar provecho a todas las posibilidades de la versión *3* de *JBuilder*, y en muchos casos habiendo instalado la actualización (los últimos parches) que sitúan el producto en una versión *3.01* (para los usuarios de *Windows*), *Borland* sorprende al mercado lanzando la versión *3.5*.

Ante tanta rapidez en la presentación de productos, cabe hacerse muchas preguntas, que se pueden resumir en las siguientes:

- ¿Incluye tantas novedades y mejoras como para constituir un nuevo producto? Y si es así...
- ¿Por qué no una versión 4?

Las respuestas a estas preguntas no son muy claras, ya que ni la gente de *marketing* de *Borland*, ni los propios ingenieros, dan contestaciones convincentes. Para los usuarios de *JBuilder 3* para *Windows*, tanto en ediciones *Standard*, *Professional* o *Enterprise*, este nuevo producto sí se puede entender como un cambio de versión, ya que se ha cambiado la filosofía del producto en ciertos aspectos, se han mejorado muchas de sus características, y como consecuencia, los fabricantes normalmente lo presentan como una nueva versión del producto.

La versión 3.5 es algo más que una simple revisión

Por otro lado, los usuarios que por ejemplo utilicen la versión para *Solaris*, apenas apreciarán cambios significativos, por lo que en este caso no tiene sentido irritar al cliente con una versión 4.

Como se aprecia, el asunto resulta algo complicado, por lo que se ha optado por lanzar el producto con la versión *3.5*, que se sitúa en un punto intermedio entre las dos posturas comentadas.

Veamos con más detalle las diferencias existentes entre la versión *3* y la *3.5*:

Para los usuarios de la edición *Foundation* de *JBuilder 3* (entorno *JBuilder* desarrollado totalmente en *Java*, la edición gratuita), la versión *3.5* sigue igual, aunque con la posibilidad de imprimir, con un programa ejecutable que elimina problemas de ejecución en *Windows*, y con la posibilidad de utilizar diferentes *JDK* (*Java Development Kit*).

Los usuarios de *JBuilder 3 Enterprise* en *Windows* y *Linux* observarán un cambio en el entorno de desarrollo, en el uso de la última versión de *VisiBroker for Java*, en la posibilidad de usar un servidor de aplicaciones, y en un nuevo control de versiones.

La edición *Estándar 3* no tiene equivalente en la versión 3.5 ya que desaparece, eso sí, pasando esta opción directamente a la edición *Foundation*, aunque con ciertas limitaciones.

La edición *Professional* para *Windows* cambia bastante, desde el entorno de desarrollo, el depurador, herramientas del entorno como *AppBrowser*, la ayuda, hasta *Swing*.

En función del entorno y de la edición, se han realizados desde mínimos cambios, hasta reestructuraciones ciertamente llamativas. A continuación, se van a detallar estos cambios, presentando así las características de *JBuilder 3.5*.

INSTALACIÓN

La instalación de *JBuilder 3.5* es una tarea muy sencilla, guiada por las instrucciones que va mostrando el asistente. Tras seleccionar unas cuantas opciones, se dispone de este nuevo entorno para comenzar a trabajar.

Por otra parte, y antes de detallar los requerimientos, conviene

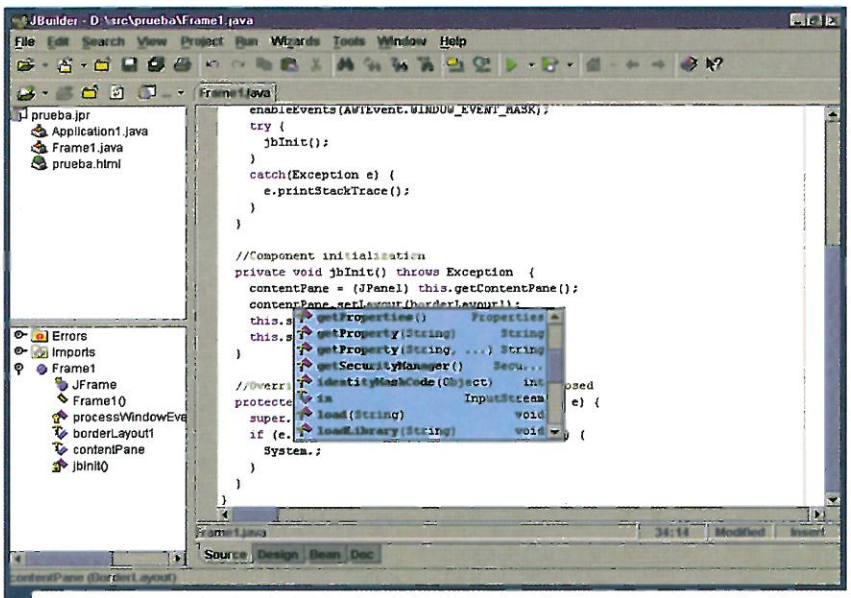


Figura 3.- Aspecto de *CodeInsight* en funcionamiento.

recordar que *JBuilder 3.5* está disponible en múltiples entornos, convirtiéndose en este aspecto, en un pionero. Así, se encuentra disponible en entornos *Windows 98* y *NT*, *Linux* para las distribuciones *RedHat 6.x*, *Mandrake 6.x* y *7.x* y en *Solaris 2.6* y *7* o 2.7.

Los requerimientos de equipo de JBuilder 3.5 resultan algo elevados

Se puede apreciar que este sí es un verdadero entorno multiplaforma, puesto que a excepción de *Mac,* el mercado de entornos *Java* esta prácticamente completo.

No se han encontrado referencias de compatibilidad con *Windows 2000,* aunque seguro que aparecen (siempre aparecerá rápidamente un *patch* en caso de existir algún tipo de problema).

Los requerimientos de equipo necesarios para el correcto funcionamiento *JBuilder 3.5* en un entorno *Windows,* son los siguientes:

- El producto indica que el procesador mínimo requerido es un *Pentium II* a 233 *Mhz* o compatible. Tras varias pruebas, se éste resulta algo justo, por lo que no es muy recomendable nada menor a 400Mhz. Ni que decir tiene, que un *Pentium III,* por ejemplo a 533Mhz, es ideal. Les aseguro que en una configuración como esta, se comporta como la seda, parece que en vez de ser un entorno para *Java* fuera un entorno clásico de *C++*.

- La *RAM* requerida es de 128 *MB*. Aunque parezca demasiado exigente, ni siquiera así trabajaremos de manera fluida. Por este motivo, conviene seguir la recomendación, puesto que con menos no se puede utilizar cómodamente, y puede llegar a desesperar. Lo ideal serían 256 *MB* , que aunque en la actualidad están algo lejos de lo que se encuentra en los entornos de desarrollo, en poco tiempo será el tamaño estándar de memoria.

- Como tarjeta de vídeo, se recomienda una *SVGA*, con una resolución mínima de 800x600, aunque se especifica que lo ideal sería 1024x768. En este aspecto, cabe indicar que una buena tarjeta de vídeo hace que todo aparente suceder de forma más fluida, por lo que es muy importante disponer de una buena tarjeta con bastante memoria, puesto que las de menos de 8 *MB* parecen no responder a la sobrecarga cuando se realizan aplicaciones con *Swing* directamente con el entorno.

- En cuanto al tamaño de disco necesario para su instalación, siguiendo la tónica general de todos los productos similares, el espacio libre requerido es de 150 *MB*.

La instalación es un proceso sencillo y rápido

Para una configuración *Linux,* los requerimientos de procesador y memoria son iguales. Conviene decir que se requieren las librerías de *glibc 2.1.1* o superior, y una versión de *kernel 2.2.5* o superior, y por supuesto, un *JDK 1.2.2*.

Para los que dispongan de una estación *Sun*, *JBuilder* requiere una *ULTRASparc 2*, y como sistema operativo, *Solaris 2.6* o *2.7*.

CARACTERÍSTICAS GENERALES

Tanto para los que empiezan a aprender *Java* como para aquellos que desean utilizar *Java 2* en sus aplicaciones, *JBuilder 3.5* ofre-

ce un potente entorno integrado para el desarrollo de aplicaciones sencillas y corporativas, desde un potente editor y un *debugger* intuitivo y sencillo de uso, hasta todas las posibilidades de *Java 2*.

El entorno de desarrollo se ha modificado, de forma que ahora cualquier tarea puede interactuar con otra, sin necesidad de minimizar ventanas ni rebuscar para encontrar una opción entre un amplio conjunto.

Integra el editor el *debugger*, los asistentes, la posibilidad del diseño gráfico directo sin necesidad de visualizar el código generado, etcétera.

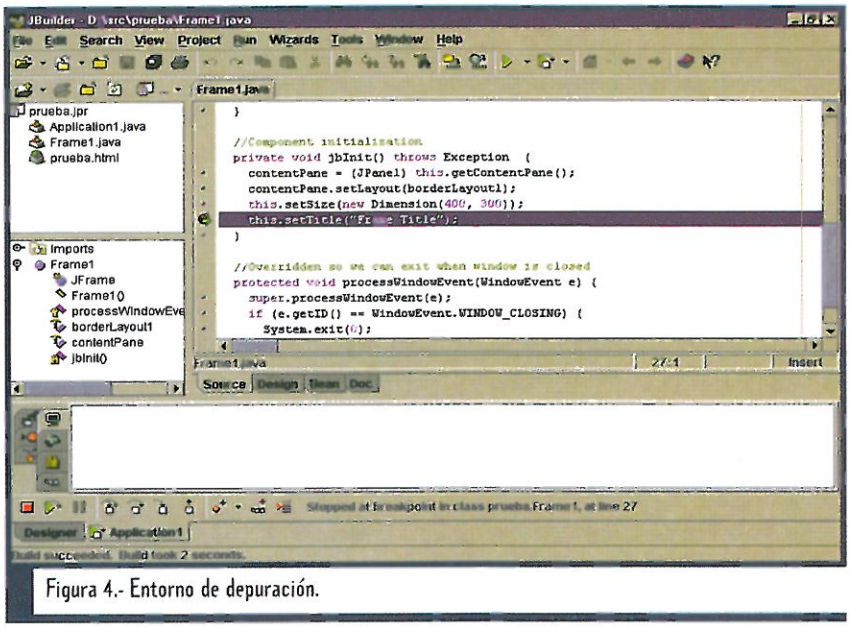


Figura 4.- Entorno de depuración.

El conjunto de asistentes ha crecido y mejorado.

Como características añadidas, permite incorporar diferentes *JDK*, herramientas de terceros, y personalizar el uso del teclado. Por ejemplo, los desarrolladores que hayan manejado *emacs* no tendrán por qué aprender otro estilo de combinación de teclas. También ofrece nuevos *JavaBeans*, y correcciones a los de anteriores versiones. En resumen, un conjunto de mejoras que hacen que el entorno sea más robusto, estable y ante todo, visualmente más sencillo de manejar, por lo que encontrar el menú o la opción necesaria requiere menos tiempo.

La herramienta integrada en el entorno, *AppBrowser*, permite la gestión de los diferentes proyectos fácilmente, posibilitando la compilación, *debugger*, gestión de clases y diseño visual entre sí.

Mediante *CodeInsight* se reducen los errores de sintaxis en el código, lo que implica un menor tiempo de desarrollo. Se ha añadido la posibilidad de acceder a las más importantes propiedades, eventos y métodos.

El *debugging* ha mejorado aún más si cabe, con el objetivo de que el desarrollador encuentre sus errores más fácilmente. Se pueden utilizar los *debugger* de los *JDK*, alterar los puntos de ruptura sin pausa, cambiar el contenido de variables y la gestión del tratamiento de los *threads* ha mejorado, evitando el cuelgue de determinado tipo de aplicaciones.

El *debugging* de aplicaciones remotas ahora se puede realizar sin necesidad de arrancar el entorno de desarrollo y se pueden utilizar otros *debugger* remotos.

La creación de *JavaBeans* mediante el asistente *BeansExpress* hace que construir *beans* sea una tarea muy sencilla, e incluso modificar los existentes, no plantea ningún tipo de problema.

Como apoyo a este proceso, se dispone de *BeanInsight* que permite diagnosticar y analizar un *bean* para obtener información sobre sus métodos, propiedades, etcétera.

Se pueden construir aplicaciones *WEB* de forma sencilla, ya que se dispone de todo lo necesario para crear *servlets*, *JSP* (*Java Server Pages*), desde asistentes, ejemplos, hasta un servidor *WEB* propio, por lo que se pueden probar incluso las aplicaciones sin salir del entorno.

JBuilder 3.5 cumple prácticamente la filosofía de no incluir ningún tipo de control en el código generado visualmente, por lo que los cambios que se hagan sobre el código generado no implicarán problemas en el aspecto visual o que el resultado no se pueda compilar.

Se ha mejorado sobre todo la presentación del código, pudiendo elegir el tipo de creación en, por ejemplo, los eventos, sin utilizar clases anónimas.

JBuilder incluye *JDataStore*, una herramienta que permite la persistencia de datos, objetos y ficheros que requieran un alto nivel de concurrencia, así como el soporte de tran-

sacciones... digamos que es el paso intermedio hasta llegar a utilizar un verdadero servidor de aplicaciones.

Puesto que *Inprise* engloba a *Borland* y *Visigenic*, con la compra de *JBuilder* podemos utilizar *Visigenic 4.0*, el *middleware* para *CORBA* que ya permite utilizar *Java 2*, y con el cual se pueden realizar aplicaciones distribuidas con mayor facilidad que en anteriores versiones.

Respecto a la conexión con bases datos, gracias a la arquitectura *DataExpress*, se pueden utilizar la mayoría de las bases de datos actuales como *Oracle*, *Sybase*, *DB2*, *Informix*, etcétera, ya que se incluye un amplio conjunto de controladores *JDBC* de distintos niveles.

Mediante asistentes, se pueden desarrollar sencillas aplicaciones estilo *Swing* que accedan a las bases de datos y muestren los contenidos y hagan búsquedas o listados de forma muy práctica.

Mediante Visigenic 4.0 se pueden crear aplicaciones CORBA

Las aplicaciones distribuidas mediante *CORBA*, *RMI* y *EJB*, se pueden desarrollar con más facilidad, por la mejora en los asistentes. Se han añadido los nuevos *CORBA-Express* y *EJBExpress*, la corrección de errores ha mejorado y un cambio en el modo de desarrollo deja claramente definido en qué estado del proceso de construcción se encuentra, de forma que una marcha atrás no crea problemas.

El conjunto de posibilidades de *JBuilder 3.5* es muy amplio, convirtiéndose en un nuevo paso tras la versión *3.0*. De esta forma, nos encontramos con un entorno al que no le falta de nada, y que seguramente será el punto de referencia para otros fabricantes por su buena integración para el desarrollo de cualquier tipo de aplicación.

EDICIONES DISPONIBLES CON JBUILDER 3.5

JBuilder 3.5 se presenta en tres ediciones diferentes, *Experta*, *Enterprise* (para empresas), y *Foundation*. Como se ha podido observar, y según se detalló anteriormente, la edición estándar ha desaparecido, integrándose en la *Foundation*. Esto ha implicado que los usuarios de esta edición pasen a utilizar una edición gratuita con apenas diferencias.

EDITION FOUNDATION

La edición *Foundation* es la más sencilla y fácil de utilizar y lo mejor de todo, es gratuita. Es cierto, ¡se descarga del *web* de *Borland* sin más!, por lo que definitivamente, es ideal para principiantes y para proyectos no muy amplios pero que requieran d toda la potencia de un entorno de desarrollo rápido del que no se necesiten mayores requerimientos, como el control de versiones, *servlets* y similares.

Con esta edición, se pueden crear *applets*, aplicaciones y *beans* mediante el uso del ratón, sin mucha necesidad de código escrito, todo ello generando código *Java 2*. Gracias a los asistentes disponibles, se puede acceder a características importantes y crear aplicaciones con *JavaBeans* mediante *JFC/Swing*, documentación mediante el uso de *JavaDoc*, *Graphics 2D*, *Collections*, etcétera.

La edición Foundation es ideal para principiantes.

Gracias a la velocidad del editor, el compilador y el *debugger*, es la herramienta ideal para iniciarse en *Java* y crear sencillas aplicaciones.

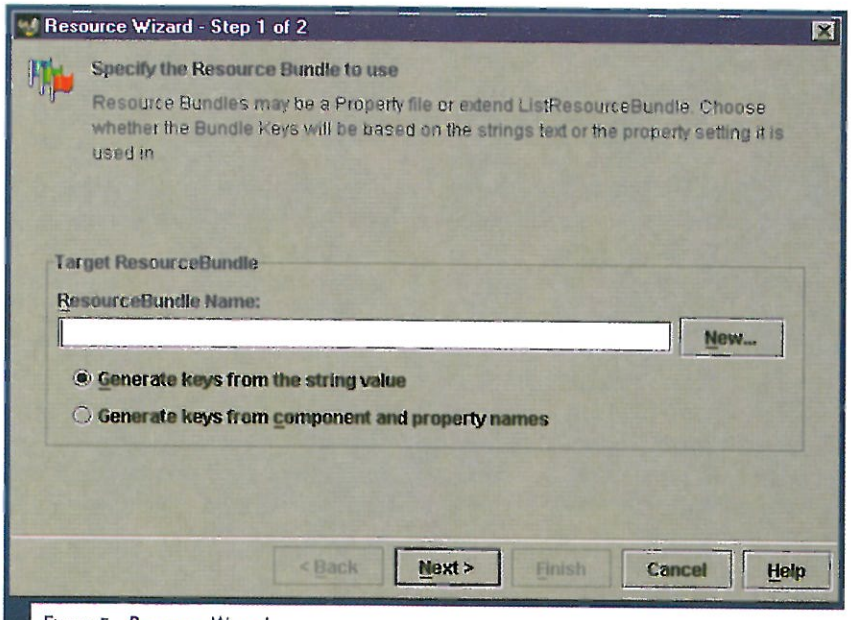


Figura 5.- *Resource Wizard.*

EDICIÓN PROFESSIONAL

Esta edición es la ideal para los programadores avanzados que prefieren un entorno desde el que controlar su aplicación sin necesidad de navegar entre mucho código generado automáticamente.

Además de todas las características de la edición *foundation*, se añaden las siguientes mejoras:

Asistentes para la creación de aplicaciones con bases de datos que generan código *Java Puro 100%*, disponiendo de una gran cantidad de controladores *JDBC*, y con unos 250 *beans* que permiten la gestión de esas bases de datos de forma sencilla y eficaz.

La edición Enterprise es el entorno más completo

El uso del asistente *DataExpress* permite utilizar los modelos de bases de datos generados por productos como *DataModeler* y generar el *SQL* adecuado.

El desarrollo de aplicaciones gráficas mediante *Swing/JFC* se ha mejorado, y el conjunto de *beans* existentes se ha optimado para una mayor velocidad de uso, ya que unas de las principales quejas de los usuarios *Java* es que las aplicaciones gráficas en determinados entornos son algo lentas y consumen mucho tiempo de proceso.

El asistente *CodeInsight* es mucho más rápido y eficaz, con total control de *inner class*. El *debugger* es más rápido, el control sobre el mismo más sencillo, y permite la gestión de *threads*, facilitando el regreso de todo un programa a un punto determinado en que se haya almacenado una pila disponible.

TABLA 1. Características de las ediciones.

Característica	Foundation	Professional	Enterprise
Mejoras en entorno	X	X	X
Intercambio JDK	X	X	X
Desarrollo JavaBeans	X	X	X
JFC/Swing	X	X	X
Asistentes, AppBrowser	X	X	X
JdataStore, CodeInsight	X	X	X
Asistentes mejorados		X	X
Internacionalización		X	X
DataStore, dbSwing		X	X
Debugging Web		X	X
Desarrollo Jsp y Servlets		X	X
Asistentes para EJB			X
Soporte RMI/CORBA			X
Debugging distribuido			X

AppBrowser se ha mejorado, el control de varios proyectos es más eficiente, y se pueden gestionar proyectos en formato *XML*.

EDICIÓN ENTERPRISE

Esta edición es la ideal para el desarrollo de las denominadas aplicaciones corporativas, que por su volumen requieran de un entorno potente que integre el máximo de posibilidades, evitando así el uso de herramientas que puedan modificar cualquier aspecto de la misma.

Es la edición más completa del mercado, ofreciendo asistentes para todas las tareas, y está especialmente pensada para crear aplicaciones distribuidas.

Además de todas las características de la edición *professional*, añade las siguientes:

Mejora sustancialmente la gestión de la persistencia mediante *JdataStore*, ya sea ésta de datos, ficheros u objetos, al utilizar automáticamente un *pool* de conexiones, optimizaciones avanzadas, etcétera.

La herramienta *DataStore Explorer* permite la gestión de las bases de datos intuitivamente.

Gracias *Visibroker for Java 4.0*, se dispone de uno de los mejores *ORB* del mercado, con lo que se pueden realizar aplicaciones *CORBA* desde el entorno de trabajo, además de crear *EJB* (*Enterprise*

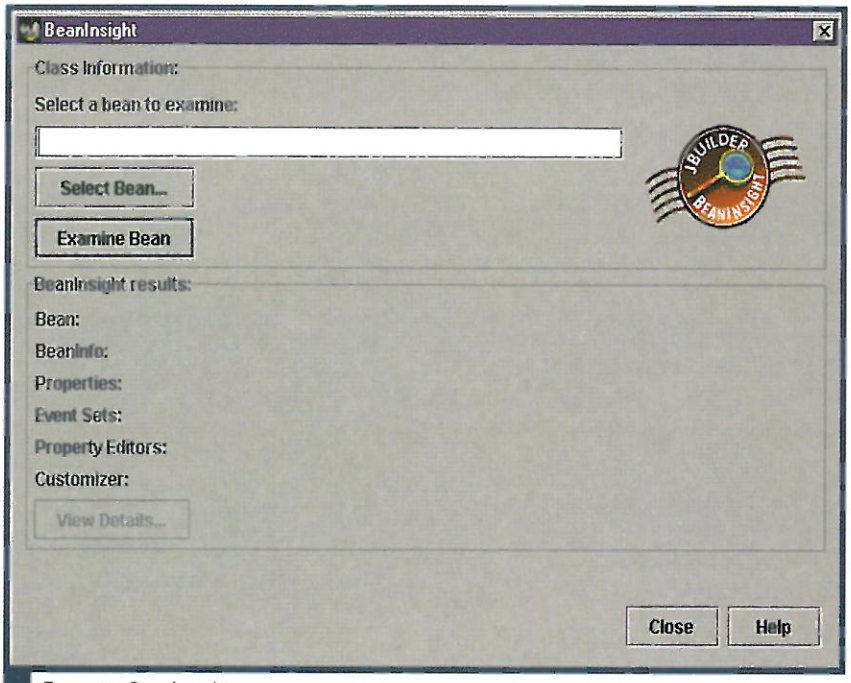


Figura 6.- *BeanInsight.*

JavaBeans) de forma más automática, robusta, sencilla y rápida, pudiendo completar cualquier desarrollo con la integración de un servidor de aplicaciones como *Inprise Application Server 4*.

Cada edición se adapta a diferentes necesidades

Se ofrecen múltiples *drivers JDBC* obteniéndose lo último de lo último en bases de datos relacionales en *Java*.

Para rematar el conjunto de posibilidades, se pueden crear aplicaciones basadas en *JSP (Java Server Pages)* y *servlets* mediante asistentes, trazas más fáciles (toda una ayuda), por lo que se pueden crear aplicaciones de *e-commerce* rápidamente.

El control de versiones mediante *CVS* es una muestra del denominado *Open Tools,* o posibilidad de integrar herramientas de terceros para ser utilizadas en el propio entorno.

Claramente, esta edición es idónea para construir aplicaciones, tanto empresariales como científicas, al ser el entorno más potente y robusto de las tres ediciones.

COMPARACIÓN DE EDICIONES

En la Tabla 1 se describen las posibilidades que ofrecen las diferentes ediciones, pudiéndose comparar fácilmente, lo que ayudará al desarrollador a decantarse por el uso de una u otra edición en función de sus necesidades presentes y futuras.

Respecto a la cuestión económica, los precios de *JBuilder* para las diferentes ediciones son: edición *Professional* 150.000 pesetas, *Enterprise* 467.500 pesetas, y la *Foundation* se puede adquirir en *CD-ROM* previo pago de 1.300 pesetas.

Las actualizaciones de versión *3* a *3.5* van desde lAS 150.000 pesetas para la versión *Enterprise* hasta 118.700 pesetas para la *Professional.*

CONCLUSIONES

Como se ha podido comprobar a lo largo de este artículo, el mundo de los entornos *Java* se mueve a una velocidad impresionante. Cuando el desarrollador está empezando a sacar provecho a un determinado entorno, aparece una nueva versión con muchas novedades que hace casi imposible no caer en la tentación del cambio.

Y en este caso, la tentación no es pequeña, puesto que el conjunto de mejoras y nuevas posibilidades es importante e interesante.

El desarrollador que haya utilizado *JBuilder* en anteriores versiones, necesitará un periodo de adaptación al nuevo entorno muy corto, y ello le permitirá realizar las tareas de forma más rápida y eficaz.

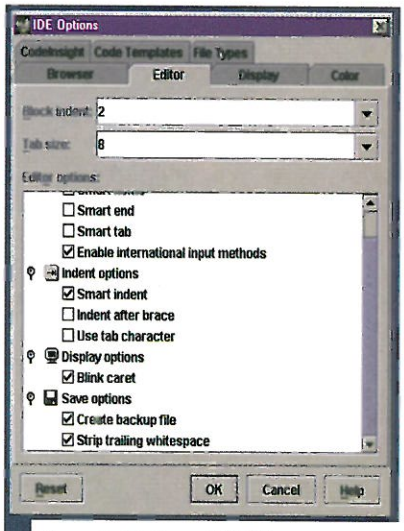


Figura 7.- Opciones del *IDE.*

SUSCRÍBETE

A

LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO

SÓLO PROGRAMADORES

QUE NO TE LÍEN CON EL <CÓDIGO>

- NOTICIAS
- MULTIMEDIA
- TEORÍA
- INTERNET
- DESARROLLO DE APLICACIONES
- COMUNICACIONES
- HERRAMIENTAS
- PROGRAMACIÓN
- ÚLTIMAS TENDENCIAS
- Y MUCHO MÁS

BOLETÍN DE SUSCRIPCIÓN

Rellene o fotocopie el cupón y envíelo a REVISTAS PROFESIONALES, S.L. (Revista Sólo Programadores).
C/ San Sotero, 5. 1ª Planta. 28037 Madrid. Tlf: 91 304 87 64. Fax: 91 327 13 03

Quiero suscribirme a la revista SÓLO PROGRAMADORES desde el Número .....
y beneficiarme de las condiciones de esta magnífica promoción:

❑ SUSCRIPCIÓN ANUAL
11 NÚMEROS + 11 CD-ROMs
AL PRECIO DE
8.580 ptas. / 51,57€

Precio de suscripción para el extranjero:
11 NÚMEROS + 11 CD-ROMs
AL PRECIO DE
11.800 Ptas.

FORMAS DE PAGO:

❑ Giro postal a nombre de REVISTAS PROFESIONALES, S.L
❑ Transferencia al Banco Popular Español. C/ Valdecanillas, 41.
Nº c/c: 0075/1040/43/ 0600047439
❑ Talón bancario a nombre de REVISTAS PROFESIONALES, S.L
❑ Domiciliación bancaria
❑ Contra reembolso

Soy antiguo suscriptor
❑ Sí ❑ No

PARA ENVÍOS AL EXTRANJERO SÓLO SE ADMITIRÁN LAS SIGUIENTES FORMAS DE PAGO:
❑ Giro postal a nombre de REVISTAS PROFESIONALES, S.L
❑ Eurocheque conformado con un banco español a nombre de REVISTAS PROFESIONALES, S.L
❑ Por VISA ____/____/____/____
Caducidad: __/__

NOMBRE Y APELLIDOS:.....

EDAD:..... PROFESIÓN: .....

TFNO: ..... DOMICILIO: .....

CIUDAD:..... C.P: .....PROVINCIA:.....

Promoción válida hasta agotar existencias

Datos de domiciliación:
Banco.....
Domicilio .....
Nº de Cuenta.....
Titular.....
Fecha .....
FIRMA

Visual C++ y MFC (IV)

Constantino Sánchez Ballesteros.
Técnico Superior de Desarrollo de Aplicaciones Informáticas.

Esta vez nos centraremos en las cajas de diálogo de *Windows* y los controles más comunes. Con estos elementos, alcanzaremos una interacción total con nuestros programas.

MENÚS

La mayoría de las aplicaciones *Windows* requieren en algún momento determinado una acción concreta del usuario, ya sea introduciendo datos, o pulsando un botón determinado. Habitualmente, la petición de datos se produce con la visualización de ventanas específicas encargadas de esta tarea, conocidas como cajas de diálogo o cuadros de diálogo. Podemos definir tres tipos para una aplicación:

- *Personalizadas*: Son aquellas que construiremos a nuestro gusto con los controles que creamos necesarios.
- *Estándar*: Son las típicas cajas de *Abrir, Archivo, Imprimir*, etcétera de *Windows*. Estas cajas ya están creadas por defecto, y no las modificaremos.
- *Predefinidas*: Estas cajas se crean por medio de la biblioteca *MFC* (por ejemplo, *MessageBox.*)

Para crear una caja de diálogo podemos utilizar dos formas:

- *Modal*: Necesita de la pulsación de un botón para poder continuar con el proceso de la aplicación. Es una caja típica cuando sale un mensaje en pantalla con los botones *Aceptar y Cancelar*.
- *No modal*: Son cajas de diálogo en las que, a diferencia de las anteriores, no es necesario cerrar para poder continuar con la ejecución de la aplicación.

CAJAS DE DIÁLOGO MODALES

Cuando queramos construir una caja de este tipo, deberemos crear un objeto de una clase derivada de *CDialog*. Para activarla, necesitaremos utilizar la función *DoModal*.

Antes de crear nuestro objeto, hay que diseñar la ventana y los controles correspondientes. Como sabemos, la ventana tiene un identificador asociado que utilizaremos para pasárselo al constructor de nuestra clase.

En el siguiente extracto de código podemos ver la activación de una ventana modal:

```
void CMiVista::OnVerDlgModal() {
   CMiDlg Dlg;
   if (Dlg.DoModal()==IDCANCEL) {
```

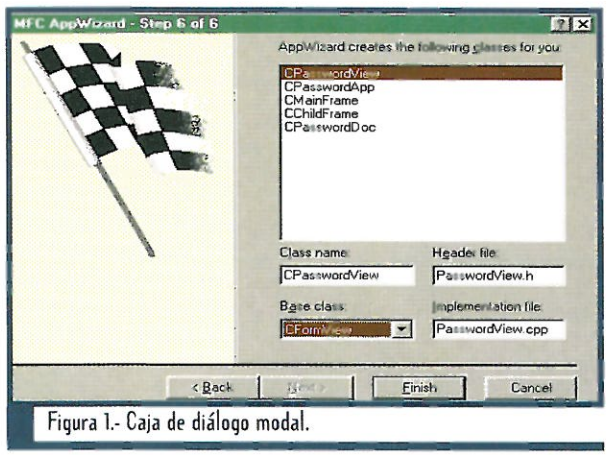


Figura 1.- Caja de diálogo modal.

```
//hacemos lo que sea...
    }
}
```

Cuando creamos una caja de diálogo, se crean dos botones (*Aceptar y Cancelar*)por omisión. Cuando se cierra una caja de diálogo ésta devolverá *IDOK* o *IDCANCEL*, dependiendo del botón que haya pulsado el usuario.

CAJAS DE DIÁLOGO NO MODALES

Para visualizar una caja de diálogo no modal tenemos que crear la plantilla en el *Editor de diálogos*, al igual que hacemos con las modales. Después hay que derivar una clase de *CDialog* para que utilice la plantilla que hemos creado en dicho editor.

Una vez hecho esto, a diferencia de las cajas de diálogo modales, necesitamos crear el diálogo. Para ello utilizaremos la función *Create* heredada de *CDialog*. Además, es necesario pasar como argumento el identificador de la plantilla que vamos a utilizar (plantilla que debemos tener creada con anterioridad.)

```
CMiDlg::CMiDlg(CWnd
     *pParent):CDialog
     (CMiDlg::IDD,pParent) {
  Create(IDD);
}
```

IDD es el identificador creado por el asistente de clases de *Visual C++* para la clase *CmiDlg*, y su valor corresponde al identificador de la plantilla de la caja de diálogo:

```
Class CMiDlg:public CDialog {
 //Construcción

public:
    CMiDlg (CWnd pParent =NULL);
    ...
    ...
}
```

El siguiente paso consistirá en definir la variable que hará referencia al diálogo que vamos a crear. Debemos definir esta variable en el lugar de la aplicación donde vayamos a utilizarla, siendo un lugar idóneo la vista de la aplicación.

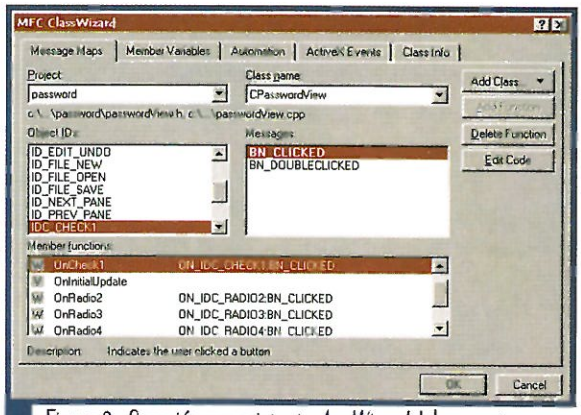


Figura 2.- Creación por asistente *AppWizard* del proyecto *Password*.

```
CMiDlg *m_pDlg;
```

Para acabar, iniciamos la variable reservando memoria para el nuevo objeto creado mediante *new*.

```
m_pDlg = new CMiDlg();
```

Para poder visualizar nuestra caja de diálogo personalizada, utilizaremos la función *ShowWindow* de *CWnd*. El siguiente extracto de código muestra cómo hacerlo:

```
m_pDlg->ShowWindow(SW_SHOW);
```

Cuando no vayamos a utilizar más nuestra caja de diálogo, es conveniente liberar la memoria utilizada mediante *delete*. Un lugar idóneo para hacerlo es en el destructor de la vista, ya que en este ejemplo es donde se definió la variable, y además es la dirección que utiliza el programa antes de terminar su ejecución.

```
void CMiVista::~CMiVista() {
   delete m_pDlg;
}
```

CAJAS DE DIÁLOGO DE ENTRADA Y SALIDA

Estas cajas son muy comunes para dar una información en pantalla, y permitir que el usuario pulse un botón cuando se haya informado del contenido del mensaje, o haya introducido un dato determinado.

Los diálogos modales requieren del usuario la pulsación de un botón para continuar

Un ejemplo muy común de este tipo de cajas de diálogo, es la ventana de introducción de contraseñas. En el mismo *Windows*, podemos ver cómo antes de entrar al sistema operativo se nos pide la introducción de una contraseña (si lo tenemos configurado de este modo.)

Mediante funciones como *Afx MessageBox* o *MessageBox*, podremos ver fácilmente cajas de diálogo de entrada y salida.

TABLA 1.

Botón Pulsado	Valor Retornado
Aceptar	*IDOK*
Cancelar	*IDCANCEL*
Anular	*IDABORT*
Reintentar	*IDRETRY*
Ignorar	*IDIGNORE*
Sí	*IDYES*
No	*IDNO*

MESSAGEBOX

Esta función, de la clase *CWnd*, permite ver una caja de diálogo de forma rápida y eficiente. Su sintaxis la presentamos en el siguiente código:

```
int MessageBox(LPCTSTR lpszText,
    LPCTSTR lpszCaption = NULL,
                                UINT
    nType = MB_OK );
```

- *lpszText*: permite introducir el texto que se visualizará en el área cliente de la caja de diálogo.
- *lpszCaption*: es el título que tendrá la ventana.
- *nType*: permite definir el número y tipo de botones, estilo del icono, y botón por omisión.

MessageBox devuelve un valor indicando el botón que ha pulsado el usuario. En la Tabla 1 podemos ver los valores posibles.

En caso de memoria insuficiente, se devolverá el valor *0*.

AfxMessageBox, es otra función general de las *MFC*, que permite realizar el mismo trabajo que *MessageBox*.

En el ejemplo siguiente se muestra cómo visualizar una caja de diálogo determinada mediante *MessageBox*:

```
::MessageBox (GetSafeHwnd(),
    "Prueba de caja de diálogo
    con
            MessageBox","PRUEBA
    DE MENSAJE", MB_ICONSTOP);
```

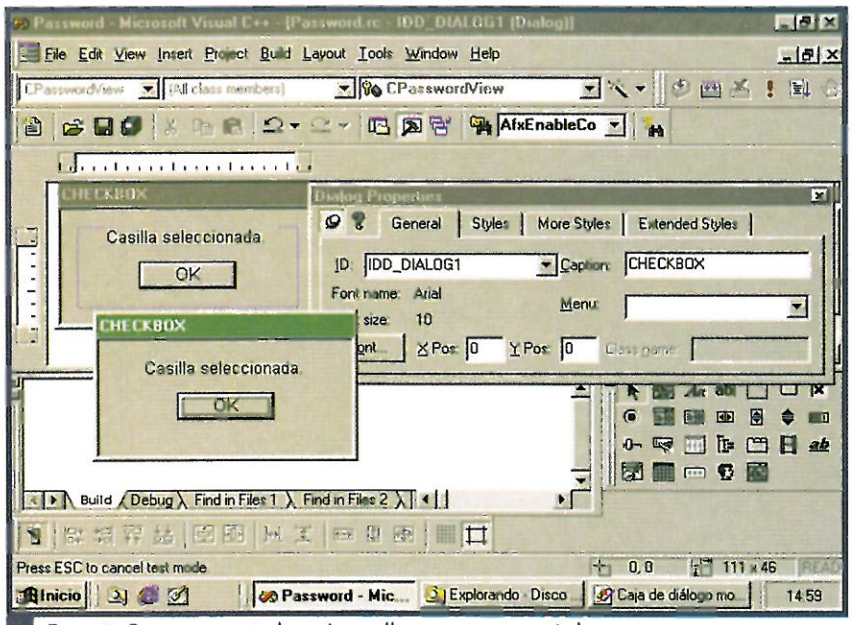


Figura 3.- Programa principal que desarrollaremos en este artículo.

PROTECCIÓN MEDIANTE CONTRASEÑA

El primer ejercicio que vamos a realizar, será la visualización de una caja de diálogo solicitando una clave antes de acceder a nuestro programa.

Para ello, crearemos un nuevo proyecto de tipo *SDI* mediante el asistente, soportando las *MFC*. A esta aplicación la llamaremos *password*. Para la clase de la vista, debemos hacer que sea una clase derivada de *CFormView*.

El objetivo de nuestra aplicación, será abrir un cuadro de diálogo que permita introducir una contraseña que dé acceso al programa principal. Para crear el cuadro de diálogo, utilizaremos el *Editor de recursos*. El nuevo diálogo debe tener activada la propiedad *System Modal*.

Tenemos que insertar un cuadro de texto que contenga los caracteres introducidos por el usuario para verificar la contraseña. Esta caja debe tener activadas (*true*) las propiedades *Password* y *Uppercase*. Con esto, logramos que nuestro diálogo sea de tipo *Modal*, que los caracteres introducidos en la caja de texto visualicen asteriscos al introducir datos, y la clave se modifique para poner todos los caracteres introducidos a mayúsculas de forma automática.

Los diálogos podemos diseñarlos desde el editor de recursos

Además, asignaremos *TAB* a *1*, para que cuando se visualice la caja de diálogo, se establezca el foco en la caja de texto. Esto lo logramos pulsando *Ctrl+D* desde el *Editor de recursos* y teniendo seleccionado el cuadro de diálogo.

Una vez creado nuestro diálogo, lo seleccionaremos y abriremos el asistente *ClassWizard*. En ese momento, se nos preguntará si

TABLA 2

CONTROL	IDENTIFICADOR	VARIABLE MIEMBRO
Cuadro combinado	(IDC_COMBO1)	m_combo
Lista	(IDC_LIST1)	m_lista
Casi. Verificación	(IDC_CHECK1)	m_casilla

LISTADO 1. Código de la función Gestiona_Contrasena

```
BOOL Gestiona_Contrasena() {

  int cont=0,boton;

  //Visualizamos caja de diálogo
  dlgpassword Dlg;

//Como mandan los cánones, sólo permitiremos 3 intentos
  while (cont<3 && (boton=Dlg.DoModal())==IDOK) {
       //la clave será soloprogramadores
    //controlamos si es la clave correcta
    if (Dlg.m_Caja !="SOLOPROGRAMADORES") {
         AfxMessageBox("Contraseña  incorrecta",MB_ICONSTOP);
         cont++;
    }
    else cont=4;
  }
  if (cont==3 || boton==IDCANCEL)
    return FALSE; //salimos
  return TRUE;
}
```

queremos crear una nueva clase para el diálogo. Aceptaremos y la llamaremos *dlgpassword*. Dejaremos el identificador asignado por defecto (*IDD_DIALOG1*).

El siguiente paso, es asignar una variable miembro de tipo *CString* a la caja de texto, para tener un acceso fácil al objeto. Esta variable la llamaremos *m_Caja*.

Cuando se inicia una aplicación, la función *InitInstance* se encarga de realizar las inicializaciones necesarias del programa. Por esto, será el lugar idóneo para llamar a la función personalizada (*Gestiona_Contrasena*) que permita el chequeo de la contraseña antes de que podamos entrar al programa principal.

La función *InitInstance* podemos encontrarla en el archivo *password.cpp*. Como se ha dicho, utilizaremos una función personalizada, a la que llamaremos para el chequeo de la contraseña. Esta función es *Gestiona_Contrasena,* y su código es el decrito en el Listado 1.

Esta función la definiremos antes de *InitInstance,* dentro del archivo *password.cpp*.

La descripción de la función es la siguiente:

- La variable *cont* servirá para contar cuántas veces se ha introducido la contraseña de forma errónea.
- *Boton* servirá para determinar qué botón de la caja de diálogo se

BUSCAMOS COLABORADORES

Si además de leer

SÓLO PROGRAMADORES

te gusta la programación, y quieres escribir en tu revista, no dudes en ponerte en contacto con nosotros. Envíanos tu propuesta junto a tu curriculum a la siguiente dirección:

REVISTAS PROFESIONALES
C/San Sotero, 5 - 1.ª planta
28037 Madrid
Ref.: Colaboraciones Sólo Programadores
E-mail: solop@virtualsw.es

ha pulsado (*Aceptar ó Cancelar = IDDOK ó IDDCANCEL.*)

- *Dlg* servirá como objeto caja de diálogo. En concreto, la que hemos creado en el *Editor de recursos*.
- Si se introduce en *m_Caja* la clave, el programa seguirá ejecutándose. En caso contrario, se retorna el valor *false* y se sale automáticamente del programa.

Para terminar, sólo nos queda hacer la llamada a la función, tal y como se muestra en el Listado 2.

CASILLAS DE VERIFICACIÓN

Dejamos momentáneamente las cajas de diálogo, para adentrarnos en los controles, comenzando por las casillas de verificación, las cuales se presentan en dos estados:

- Activadas.
- Desactivadas.

Las casillas de verificación y botones de opción sólo pueden tener dos estados

Cada casilla de verificación es independiente de las demás, puesto que cada una tiene un identificador diferente. Cuando queramos saber si una casilla está activada o no, debemos verificar el valor devuelto por la función *GetCheck* de la clase *CButton*. Los valores devueltos pueden ser los siguientes:

- *0*: Casilla vacía (no seleccionada).
- *1*: Casilla seleccionada, marcada con ÷.
- 2: Casilla deshabilitada, de color gris.

El valor *2* sólo estará presente si durante el diseño ponemos la propiedad *tri-state* de la casilla en *true* (activada). Para deshabilitar en modo de diseño una casilla, debemos poner la propiedad *Disabled* en *true*.

BOTONES DE OPCIÓN

Un botón de opción indica si una opción determinada está activada o no. Al igual que las casillas de verificación, cada botón es independiente, tiene sus propios identificadores.

Normalmente, un botón de opción forma parte de un grupo. Este grupo es el que contendrá un número determinado de botones de opción. De este modo, si se selecciona uno de los botones del grupo, los demás quedarán desactivados.

Para saber si un determinado botón está seleccionado, debemos consultarlo a la función *GetCheck* de la clase *CButton*. Los valores que devuelve la función son los siguientes:

- *0:* si el círculo que representa al botón está desactivado.
- *1:* si el círculo aparece rellenado con un punto negro (•).

LISTADO 2. Llamada a la función.

```
BOOL CPasswordApp::InitInstance() {
   ////////////////////////////////////////////
   if (!Gestiona_Contrasena())
         return FALSE;
   ////////////////////////////////////////////

   AfxEnableControlContainer();

   // Standard initialization
// If you are not using these features and wish to //reduce the size
//  of your final executable, you should remove from //the following
//  the specific initialization routines you do not //need.

#ifdef _AFXDLL
Enable3dControls();
// Call this when using MFC //in a shared DLL
#else
Enable3dControlsStatic();
// Call this when //linking to MFC //statically
#endif
...
...
}
```

Cuando se pulsa sobre un botón de opción, *Windows* envía el mensaje de notificación *BN_CLICKED* a la ventana padre del botón.

GRUPOS

Los grupos suelen contener conjuntos de botones de opción o casillas de verificación. Un grupo funciona del siguiente modo: cuando se selecciona uno de sus elementos, los demás se desactivan de forma automática.

Los cuadros combinados se forman con un cuadro de texto y una lista

Si colocamos botones de opción o casillas directamente en una ventana, formarán un único grupo. Para crear diferentes grupos debemos utilizar la propiedad *Group*. Un grupo de botones se forma siguiendo el orden de tabulación, desde el primer botón que tenga activada la propiedad *true* hasta el último que también la tenga.

LISTAS

Las listas permiten seleccionar un elemento entre varios disponibles. Cuando haya un número de elementos que sobrepase físicamente la capacidad del contenedor de lista, aparecerá automáticamente una barra de desplazamiento vertical para poder ver los elementos restantes.

Es posible ver los elementos en una única columna, o en varias. Existen dos variantes de listas:

- Las listas propiamente dichas.
- Los cuadros combinados.

Los cuadros combinados son una combinación de caja de texto y lista. Se pueden apreciar, cuando en los programas vemos una caja de texto con un botón a su derecha, y al pulsarlo permite desplegar una lista con un conjunto de elementos. Estos elementos se pueden seleccionar directamente desde esa lista o escribiéndolos en la caja de texto.

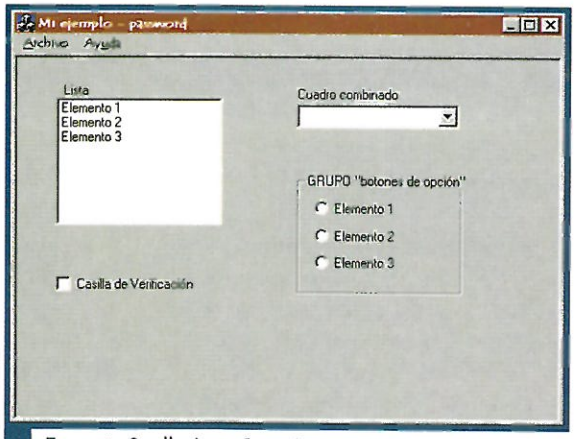


Figura 4.- Casilla de verificación.

Por el contrario, en una lista, sólo podemos seleccionar un elemento de forma interactiva mediante el ratón, ya que no es posible escribir en la lista.

Para insertar elementos en una lista o cuadro combinado en tiempo de ejecución, debemos utilizar las funciones *AddString* e *InsertString* de las clases *CListBox* y *CComboBox*. Para la supresión de elementos se utiliza la función *DeleteString*. Existe una función especial, *ResetContent*, que permite borrar toda la lista.

AGREGANDO CONTROLES

Para completar el ejemplo de la contraseña, vamos a implementar los controles vistos hasta ahora:

- Grupo.
- Lista.
- Cuadro combinado.
- Casilla de verificación.
- Botones de opción.

El funcionamiento del programa será el siguiente: Aparte de la funcionalidad establecida hasta ahora (petición de contraseña), insertaremos una lista con tres elementos, los cuales podremos seleccionar pulsando sobre la lista o activando el botón de opción correspondiente.

Además, al activar la casilla de verificación, se despliega de forma automática el cuadro combinado, mostrando los dos elementos insertados. Uno de ellos se ha insertado en tiempo de diseño, y otro en tiempo de ejecución.

Lo primero que tenemos que hacer es insertar los controles necesarios. Para ello, vamos al *Editor de recursos*, activamos el diálogo principal del programa (*IDD_PASSWORD_FORM*) e insertamos los controles para que aparezcan tal y como se muestran en la imagen.

Dejaremos como identificadores de los controles los que asigne de forma automática *Visual C++*. Lo único que debemos tener en cuenta es la creación de los botones de opción. Para que funcionen de forma agrupada, debemos activar sólo en el primer botón de opción la propiedad *Group*. En los otros dos ésta quedará desactivada.

Hay que tener en cuenta que cuando se forman nuevos grupos, siempre se activa la propiedad *Group* en el primer miembro del grupo, y los demás deben tener desactivada esta propiedad.

El siguiente paso es definir variables miembro para los siguientes controles, recordando que se hace mediante *ClassWizard* (ver Tabla 2.)

Con estas variables, no tendremos que definir nuevos objetos para los controles, lograremos un fácil acceso desde la vista a los métodos y propiedades de cada control. Una vez definidas las variables de los controles, pasaremos a establecer la funcionalidad buscada en el programa. Lo primero es agregar elementos a las listas.

Cuando trabajamos con MFCs, el uso de Classwizard se hace indispensable

El evento de la vista *OnInitial Update* es idóneo para realizar esta tarea, ya que este evento se ejecuta al actualizar por primera vez la ventana del programa.

```
void
    CPasswordView::OnInitialUpdat
    e() {
    CFormView::OnInitialUpdate();
    GetParentFrame()-
    >RecalcLayout();
    ResizeParentToFit();
    GetParent()->CenterWindow();
    //listas y combos
    m_combo.AddString
    ("Lectores");
    m_lista.AddString ("Elemento
    1");
    m_lista.AddString ("Elemento
    2");
    m_lista.AddString ("Elemento
    3");
}
```

Mediante *AddString* agregamos las cadenas de caracteres que precisemos (elementos), tanto a la lista como al cuadro combinado (*combo.*)

Cuando pulsamos con el botón del ratón sobre una casilla o botón de opción, *Windows* envía el mensaje *BN_CLICKED*. Este mensaje lo aprovecharemos para ejecutar las instrucciones que precisemos en ese momento dado.

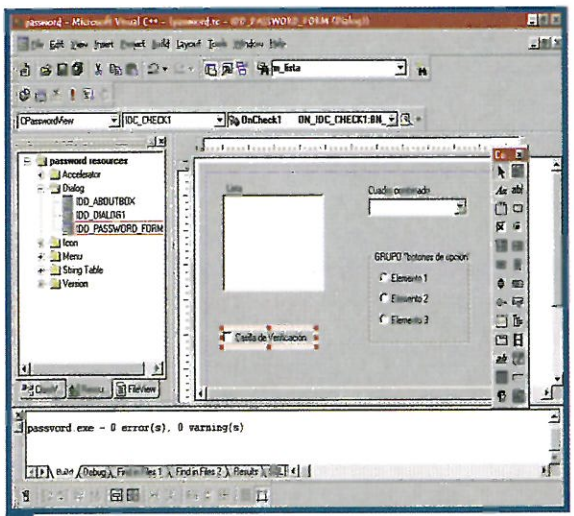


Figura 4.- Casilla de verificación.

Nosotros utilizaremos este evento para la casilla de verificación y para los tres botones de opción. Mediante *ClassWizard* sólo tenemos que seleccionar los identificadores de estos elementos y agregarles el evento *BN_CLICKED*.

Cuando se active la casilla de verificación, haremos que se despliegue el cuadro combinado con los elementos que tenga:

```
void CPasswordView::OnCheck1() {
    if (m_casilla.GetCheck ())
     m_combo.ShowDropDown (true);
    else
    MessageBox("Casilla desactiva-
     da","CHECKBOX",MB_OK);
}
```

Si se desactiva la casilla aparecerá un mensaje (diálogo) avisándonos del estado de la casilla. Para desplegar un cuadro combinado utilizaremos el método *ShowDropDown*.

Cuando se pulse uno de los botones de opción que representan elementos, haremos que se seleccione de forma automática el elemento correspondiente en la lista.

```
void CPasswordView::OnRadio2() {
    m_lista.SetCurSel (0);
}

void CPasswordView::OnRadio3() {
    m_lista.SetCurSel (1);
}

void CPasswordView::OnRadio4() {
    m_lista.SetCurSel (2);
}
```

En el método *SetCurSel* debemos asignar el valor correspondiente al elemento de la lista que queremos seleccionar. Hay que tener en cuenta que el primer elemento comienza en el índice *0*.

CONCLUSIÓN

Termina aquí la serie de artículos dedicados a *Visual C++*. Esperamos que os haya sido de utilidad, y que pueda serviros de ayuda para vuestros desarrollos presentes y futuros. Os recordamos que, como siempre, en el *CD-ROM* encontraréis toda la documentación complementaria referida al artículo.

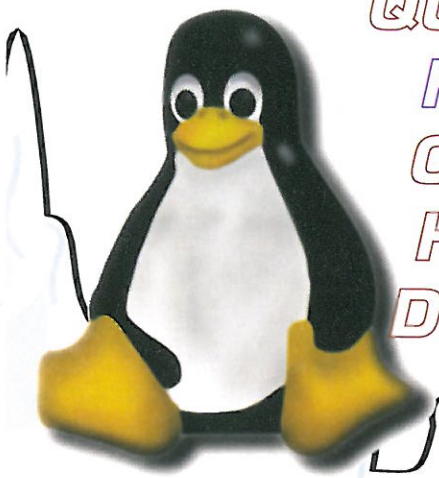
SUSCRIPCIÓN DOBLE

LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO

SÓLO PROGRAMADORES

Sólo PROGRAMADORES LINUX

QUE NO TE LÍEN CON EL <CÓDIGO>



PARA NO QUEDARSE HELADO CUANDO HABLEN DE LINUX

BOLETÍN DE SUSCRIPCIÓN

Rellene o fotocopie el cupón y envíelo a REVISTAS PROFESIONALES, S.L. (Revista Sólo Programadores).
C/ San Sotero, 5. 1ª Planta. 28037 Madrid. Tlf: 91 304 87 64. Fax: 91 327 13 03

Quiero suscribirme a la revistas SÓLO PROGRAMADORES y SÓLO PROGRAMADORES LINUX desde el Número .....
y beneficiarme de las condiciones de estas magníficas promociones:

❑ **SUSCRIPCIÓN ANUAL**
22 NÚMEROS + 22 CD-ROMs
AL PRECIO DE
14.822 ptas. / 89,09€

Precio de suscripción para el extranjero:
22 NÚMEROS + 22 CD-ROMs
AL PRECIO DE
21.200 Ptas.

FORMAS DE PAGO:

❑ Giro postal a nombre de REVISTAS PROFESIONALES, S.L
❑ Transferencia al Banco Popular Español. C/ Valdecanillas, 41.
Nº c/c: 0075/1040/43/ 0600047439
❑ Talón bancario a nombre de REVISTAS PROFESIONALES, S.L
❑ Domiciliación bancaria
❑ Contra reembolso

Soy antiguo suscriptor
❑ Sí ❑ No

PARA ENVÍOS AL EXTRANJERO SÓLO SE ADMITIRÁN LAS SIGUIENTES FORMAS DE PAGO:
❑ Giro postal a nombre de: REVISTAS PROFESIONALES, S.L
❑ Talón conformado en dólares a nombre de: REVISTAS PROFESIONALES, S.L
❑ Por VISA ___/___/___/___
Caducidad: __/__

NOMBRE Y APELLIDOS:.....

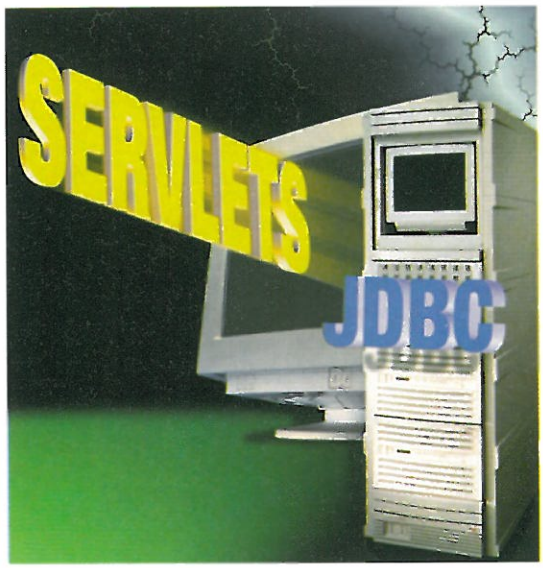
EDAD:..... PROFESIÓN: .....

TFNO: ..... DOMICILIO: .....

CIUDAD:..... C.P: .....PROVINCIA:.....

Promoción válida hasta agotar existencias

Datos de domiciliación:
Banco.....
Domicilio .....
Nº de Cuenta.....
Titular.....
Fecha .....
FIRMA



Aplicaciones Web con acceso a BBDD basadas en Java (y V)

Adolfo Aladro García. *Analista Programador*

La nueva versión de la *API JDBC* incluida en *JDK 1.2*, aporta mejoras en el rendimiento de acceso a bases de datos, y facilitan la labor de los desarrolladores de aplicaciones. Vamos a analizar las más interesantes.

RENDIMIENTO DE LAS APLICACIONES WEB

En las entregas anteriores hemos analizado el acceso a bases de datos desde los *servlets* desde el punto de vista de la gestión de las conexiones. Aunque se trata de un factor determinante del rendimiento de las aplicaciones *WEB*, no lo es menos la forma en la que se accede y se manipulan los datos obtenidos.

La *API JDBC 1.0* impone serias itaciones, puesto que sólo se de acceder secuencialmente a la información almacenada por un objeto *ResultSet*, registro a registro, desde el principio hasta el fin. En aplicaciones complejas, a veces no queda más remedio que almacenar los datos de un objeto *ResultSet* en una estructura temporal (*Vector, Hashtable*, etc.)

La API JDBC 2.0 resulta muy flexible

Posteriormente se accede al medio de almacenamiento secundario de la manera más conveniente. Esto, que puede ser una solución para muchos problemas, no deja de representar un desperdicio de recursos. Se incrementa el uso de memoria de la aplicación, y en general crece también el tiempo de procesamiento de las peticiones. Todos estos factores suelen disminuir el rendimiento de las aplicaciones *WEB*.

La *API JDBC 2.0* solucionará muchos de estos problemas, ya que los objetos *ResultSet* son mucho más flexibles que los de la versión anterior. Esto evita el uso (y abuso) de mecanismos de almacenamiento intermedio. Por otro lado, la *API* también ofrece formas para realizar actualizaciones, inserciones y borrados de registros de forma mucho más cómoda y eficiente.

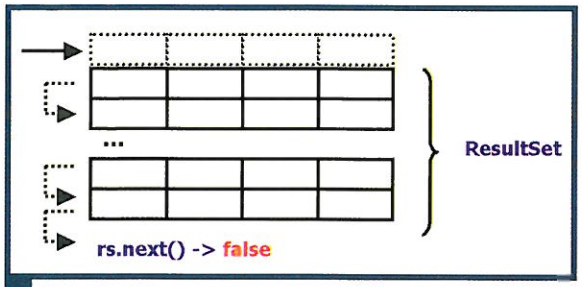


Figura 1.- Esquema del movimiento que hace el puntero de un objeto *ResultSet* en el *API JDBC* 1.0.

MOVIMIENTO DE CURSORES

A lo largo de los ejemplos de esta serie, los resultados devueltos por la ejecución de una sentencia *SQL* se recorren de manera secuencial, registro a registro desde el primero hasta el último. Así, para obtener la lista de todos los discos de la base de datos, ordenados por el nombre del artista, utilizaremos el código siguiente:

```
Statement s = c.createState-
    ment();
ResultSet rs =
    s.executeQuery("SELECT * FROM
    DISCO ORDER BY
    DISCO.DIS_ARTISTA");
boolean seguir = rs.next();
while (seguir) {
  String artista = rs.getS-
    tring("DIS_ARTISTA");
  ...
  seguir = rs.next();
}
rs.close();
s.close();
```

La sentencia *Select* devuelve un conjunto de registros de la base de datos. En ese mismo instante, el puntero del objeto *ResultSet* está por defecto apuntando a la posición inmediatamente anterior a la que ocupa el primer registro. A medida que se invoca al método *next*, el puntero avanza hasta llegar al final. La variable *seguir* toma el valor *false* cuando no quedan más registros que recorrer.

Esta forma de acceder a los datos obtenidos en una consulta *SQL* es muy rígida. Por este motivo, la *API JDBC 2.0* ha propuesto un nuevo modelo de objeto *ResultSet*, en el que la forma de acceso a la información es mucho más versátil y potente. Así, el cursor que recorre la tabla de registros podrá moverse hacia delante y hacia atrás, o ir a una posición determinada. En definitiva, la posición del cursor podrá ser comprobada y establecida en cualquier momento.

La forma en la que se indica el tipo de objeto *ResultSet*, viene determinada por la llamada al método *createStatement* de la interfaz *Connection*. En la *API JDBC 2.0*, éste admite dos parámetros:

- `public Statement createState-
  ment(int resultSetType, int
  resultSetConcurrency)
throws SQLException`

El primero de ellos indica el tipo de objeto *ResultSet* que será utilizado para guardar los datos procedentes de ejecutar la consulta *SQL*. Puede tener tres valores:

- `TYPE_FORWARD_ONLY`

El cursor solamente puede moverse hacia delante, tal y como ocurre en el objeto *ResultSet* de la *API JDBC 1.0*.

- `TYPE_SCROLL_INSENSITIVE` ó `TYPE_SCROLL_SENSITIVE`

El cursor puede moverse libremente, recorriendo así los datos almacenados en el objeto *ResultSet* de la manera que se considere más conveniente.

El tipo *TYPE_SCROLL_INSENSITIVE*, indica que el cursor no se ve afectado por los cambios que pueda haber en el objeto *ResultSet* mientras el cursor está siendo utilizado. Por el contrario, el tipo *TYPE_SCROLL_SENSITIVE* indica que el cursor "verá" todos las modificaciones que puedan producirse.

Con TYPE_SCROLL_SENSITIVE el cursor puede moverse en todas direcciones

En lo anterior, encontramos un nuevo aspecto de los objetos *ResultSet* de la *API JDBC 2.0* que no tiene que ver con el movimiento de los cursores: los objetos *ResultSet* pueden ser modificados, y con ellos, y en última instancia, la base de datos. Por el momento, es suficiente que conozcamos los dos valores que puede tomar el segundo de los parámetros del método *createStatement*.

- `CONCUR_READ_ONLY`

El objeto *ResultSet* obtenido como consecuencia de ejecutar una sentencia *SQL*, no puede ser modificado.

- `CONCUR_UPDATABLE`

En este caso, el objeto *ResultSet* puede ser modificado, y con ello la base de datos.

Una sentencia *SQL* creada de la siguiente manera:

```
Statement s =
    c.createStatement(ResultSet.T
    YPE_FORWARD_ONLY,
                  ResultSet.CON-
    CUR_READ_ONLY);
```

Dará lugar a un objeto *ResultSet* idéntico al que utiliza la *API JDBC*

1.0 (obsérvese que la llamada al método *createStatement* no recibe ningún parámetro en la versión *1.0* de la *API JDBC*).

NUEVOS MÉTODOS

Ya conocíamos el método *next* para recorrer los registros de un objeto *ResultSet*. Este método se completa ahora con un amplio conjunto de otros métodos, que permiten llevar a la práctica las distintas alternativas que propone la *API JDBC 2.0* para mover los cursores. Para empezar, *previous* es el método complementario y se utiliza para desplazar el cursor una posición atrás.

El método previous permite desplazar el cursor hacia atrás

- `public boolean previous() throws SQLException`

De la misma manera que pasaba con el método *next*, *previous* devuelve *false* cuando se intenta acceder a un registro que no existe.

Los registros de un objeto *ResultSet* pueden ser obtenidos directamente mediante la utilización de los siguientes métodos:

- `public boolean first() throws SQLException`

Desplaza el cursor al primer registro del objeto *ResultSet*.

- `public void beforeFirst() throws SQLException`

Sitúa el cursor en la posición inmediatamente anterior a la correspondiente al primero de los registros (ésta es la posición en la que se encuentra por defecto el cursor, justo después de que se haya creado el objeto *ResultSet* como consecuencia de la ejecución de una sentencia *SQL*).

- `public boolean last() throws SQLException`

El cursor se mueve hasta apuntar al último de los registros.

- `public void afterLast() throws SQLException`

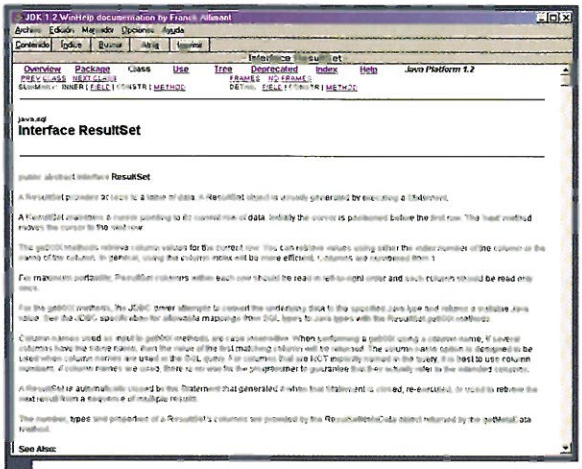


Figura 2.- Ayuda del *JDK* 1.2 donde se relatan las novedades del *JDBC* 2.0 con respecto a los objetos del tipo *ResultSet*.

Desplaza el cursor a la posición inmediatamente posterior a la correspondiente al último de los registros.

El ejemplo anterior se presenta para la *API JDBC 2.0* de la forma siguiente:

```
Statement s =
     c.createStatement(ResultSet.T
     YPE_SCROLL_INSENSITIVE,
                       ResultSet.CON-
     CUR_READ_ONLY);
ResultSet rs =
     s.executeQuery("SELECT * FROM
     DISCO
       ORDER BY DISCO.DIS_ARTIS-
     TA");
rs.afterLast();
boolean seguir = rs.previous();
while (seguir) {
  String artista = rs.getS-
     tring("DIS_ARTISTA");
  ...
  seguir = rs.previous();
}
rs.close();
s.close();
```

La llamada al método *afterLast* coloca el cursor un registro después del último de los registros. En cada iteración del bucle, el cursor se desplaza una posición atrás mediante la invocación del método *previous*.

Otro de los métodos que pueden utilizarse es *absolute*, el cual permite colocar el cursor en una posición específica dentro del conjunto de registros almacenados por el objeto *ResultSet*.

- `public boolean absolute(int row) throws SQLException`

Recibe como parámetro el número correspondiente al registro al que deseamos que apunte el cursor. Si el número es positivo, el cursor se mueve hasta señalar al registro que se encuentra en esa posición, teniendo en cuenta que el primero de los registros es el *1*.

La acción rs.first() equivale a rs.absolute(1)

Si el número es negativo, el cursor se mueve a la posición *n+1-row*, dónde *n* es el número total de registros almacenados por el objeto *ResultSet*. Así, por ejemplo, tenemos que *rs.absolute(-1)* es equivalente a hacer *rs.last()*, y *rs.absolute(1)* es lo mismo que hacer *rs.first()*.

Hasta ahora, hemos visto aquellos métodos que sirven para mover

el cursor. También existen otros métodos para saber cuál es su posición en un momento dado. Entre ellos está el método *getRow*.

```
• public int getRow() throws
  SQLException
```

Este método devuelve la posición a la que está apuntando el cursor en cada momento. Otros métodos son:

```
• public boolean isAfterLast()
  throws SQLException
```

La API JDBC 2.0 resulta muy flexible

Devuelve *true*, en el caso de que el cursor se encuentre apuntando a la posición inmediatamente posterior a la posición correspondiente al último de los registros del objeto *ResultSet*.

```
• public boolean isFirst()
  throws SQLException
```

Devuelve *true* si el cursor señala al primero de los registros.

```
• public boolean isLast()
  throws SQLException
```

Devuelve *true* si el cursor señala al último de los registros.

ACTUALIZACIÓN MEDIANTE OBJETOS RESULTSET

Una de las novedades más interesantes de la *API JDBC 2.0*, es la posibilidad de actualizar la base de datos mediante la utilización directa de objetos *ResultSet*.

Para que los datos de un objeto *ResultSet* puedan ser modificados, la sentencia *SQL* a partir de la cual fue creado tiene que ser creada con la variable *ResultSet.CONCUR_UPDATABLE*.

Con la *API JDBC 1.0* las actualizaciones de la base de datos conllevan siempre la utilización de una sentencia *SQL* de tipo *update*. Por ejemplo:

```
int rs = s.executeUpda-
    te("UPDATE DISCO SET
    DISCO.DIS_TITULO
    = 'Burning down the house'
    WHERE
    DISCO.DIS_ID_DISCO=10");
```

Como alternativa, la *API JDBC 2.0* permite realizar la misma operación, actuando directamente sobre un objeto *ResultSet*. Para ello, la interfaz *ResultSet* proporciona una serie de métodos *updateXXX* que actúan sobre el registro al que apunta el cursor en cada momento. Algunos ejemplos de éstos son:

```
• public void updateBoolean(int
  columnIndex, boolean x)
  throws SQLException
• public void updateDate(int
  columnIndex, Date x) throws
  SQLException
• public void updateString(int
  columnIndex, String x) throws
  SQLException
• public void updateInt(int
  columnIndex, int x) throws
  SQLException
...
```

El primero de los parámetros representa la columna, dentro de la fila actual, que va a ser actualizada. Puede ser un número entero, tal y como vemos en las definiciones anteriores, o bien una cadena con el nombre del campo. Por ejemplo:

```
• public void
  updateBoolean(String column-
  Name, boolean x) throws
  SQLException
• public void updateDate(String
  columnName, Date x) throws
  SQLException
• public void
  updateString(String columnNa-
  me, String x) throws SQLEx-
  ception
• public void updateInt(String
  columnName, int x) throws
  SQLException
...
```

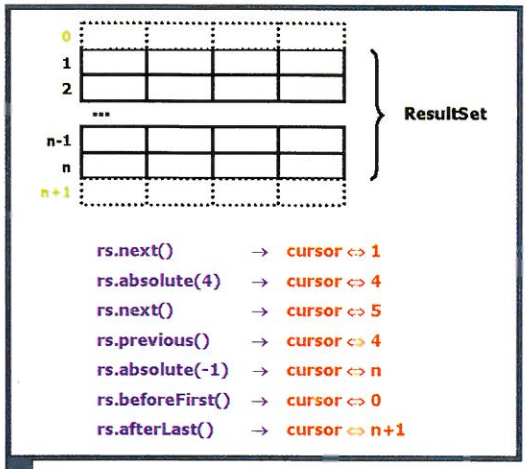


Figura 3.- Ejemplo de utilización de los métodos que permiten recorrer el conjunto de resultados en todas direcciones.

JDBC 2.0 permite actualizar mediante ResultSet

Teniendo en cuenta esto, los procesos de actualización pueden hacerse actuando sobre los objetos *ResultSet*: primero se mueve el cursor hasta que apunte al registro deseado, y posteriormente se invoca un método *updateXXX*. Por último, el método *updateRow* guarda la información actualizada en el objeto *ResultSet*, así como en la base de datos.

```
rs.absolute(5);
rs.updateString("DIS_TITULO","Bur-
    ning down the house");
rs.updateRow();
```

El método InsertRow concluye la inserción

El método *updateRow* es el que marca la actualización real de la base de datos, y si no es invocado, los campos no son actualizados.

- `public void updateRow()`
  `throws SQLException`

INSERCIÓN Y ELIMINACIÓN

Acabamos de ver cómo se pueden actualizar registros de la base de datos utilizando directamente los métodos que ofrece la *API JDBC 2.0* en vez hacerlo mediante sentencias *UPDATE*. Esto mismo puede ser usado para insertar y eliminar registros de la base de datos.

El método *moveToInsertRow* de la interfaz *ResultSet* desplaza el cursor de manera que apunta a una fila inexistente, que será utilizada para llevar a cabo la inserción.

- `public void moveToInsertRow()`
  `throws SQLException`

En este momento deben invocarse los mismos métodos *updateXXX* que se han descrito en el apartado anterior. Finalmente, para completar la inserción del nuevo registro, debe ser llamado el método *insertRow*.

- `public void insertRow()`
  `throws SQLException`

Así, por ejemplo, podríamos hacer una inserción en la tabla *Disco* de nuestro ejemplo mediante los siguientes pasos:

```
rs.moveToInsertRow();
rs.updateString("DIS_TITULO", "The
    Next Best Thing");
rs.updateString("DIS_ARTISTA",
    "Varios");
rs.updateString("DIS_FECHA",
    "2000");
rs.insertRow();
```

Cuando se termina de ejecutar la sentencia *insertRow*, el registro correspondiente es insertado, tanto en la base de datos, como en el objeto *ResultSet*.

Para eliminar registros de la base de datos, el proceso es muy similar.

- `public void deleteRow()`
  `throws SQLException`

En primer lugar hay que desplazar el cursor hasta que apunte al registro que se desea borrar. Posteriormente se invoca al método *deleteRow*.

Hay que tener en cuenta que todas estas características de la *API JDBC 2.0* son muy dependientes del controlador que se esté utilizando.

Hay casos en los que el controlador no permite ciertas operaciones, y otros en los que simplemente varía la gestión de las filas eliminadas. Por todo ello, es recomendable consultar la documentación del controlador antes de empezar a utilizar el *API JDBC 2.0*.

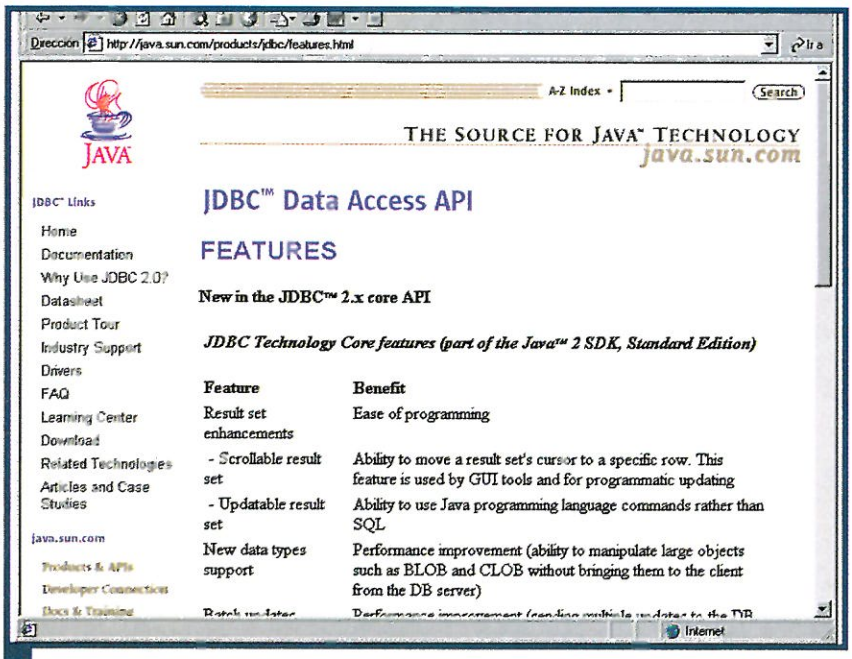
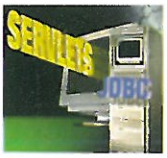


Figura 4.- Página *Web* de *Sun*, donde se detallan algunas de las características más señaladas del *API JBDC* 2.0.

ACTUALIZACIONES POR LOTES

En la *API JDBC 1.0* todas las sentencias *SQL* son tratadas como entidades individuales, y como tales son enviadas a la base de datos. Esto significa, que para ejecutar varias sentencias *SQL* de actualización, será preciso crear tantas sentencias como sea necesario, e invocar al método *executeUpdate* con cada una de ellas.



Contrastando con lo anterior, la *API JDBC 2.0* permite tener un *buffer* de sentencias *SQL* que pueden ser mandadas a la base de datos como un lote. Para ello, la interfaz *Statement* cuenta con el método *addBatch*.

- `public void addBatch(String sql) throws SQLException`

Este método tiene un único parámetro, que se corresponde con la sentencia *SQL* que se desea añadir al *buffer* de sentencias asociado con ese objeto *Statement*. Veamos un ejemplo:

```
c.setAutoCommit(false);
Statement s = c.createState-
    ment();
s.addBatch("INSERT INTO DISCO
    (DIS_ARTISTA, DIS_TITULO,
    DIS_FECHA)
     VALUES('Day One', 'Ordinary
    Man', '2000')");
s.addBatch("INSERT INTO DISCO
    (DIS_ARTISTA, DIS_TITULO,
    DIS_FECHA)
    VALUES('William Orbit', 'Pie-
    ces in modern style',
    '2000')");
...
int[] rs = s.executeBatch();
```

En primer lugar, se llama al método *setAutoCommit* de la interfaz *Connection*. El valor del parámetro es *false*, lo que significa que las sentencias *SQL* que se lancen contra esa conexión serán tratadas como un todo o se validan todas, trasladándose el resultado a la base de datos de manera definitiva, o los cambios de todas ellas son deshechos.

Los objetos Statement tienen asociado un buffer de sentencias

El segundo paso consiste en ir llenando el *buffer* asociado al objeto *Statement* con todas aquellas sentencias *INSERT* que sean necesarias. Por último, se llama al método *executeBatch*:

- `public int[] executeBatch() throws SQLException`

El método *executeBatch* hace que la base de datos reciba todas las sentencias *INSERT* de manera que éstas sean tratadas como un todo. El resultado de este método es un *array* de números enteros, cada uno de los cuales se corresponde con los resultados de ejecutar la sentencia *SQL* que se encuentra en la misma posición dentro del *buffer*.

Si por algún motivo, los resultados de ejecutar esas sentencias no fuera el esperado, podrían deshacerse los cambios que hasta el momento se hubieran hecho en la base de datos simplemente haciendo:

```
try {
  c.setAutoCommit(false);
  ...
} catch (SQLException e) {
  c.rollback();
  c.close();
}
```

CONCLUSIÓN

La *API JDBC 2.0* introduce un considerable número de características nuevas que permiten mejorar la eficiencia de las aplicaciones que acceden a base de datos. Sin embargo, no debemos olvidar que algunas de estas características son dependientes del controlador *JDBC* que se utilice.

Antes de pasar a utilizar todos estos aspectos de la *API JDBC 2.0*, es necesario comprobar que la versión que tenemos es la adecuada, y además es preciso tener garantías de que el controlador *JDBC* concreto con el que trabajamos los soporta. Este es quizás el mayor inconveniente que presenta esta nueva versión de la *API JDBC 2.0*.

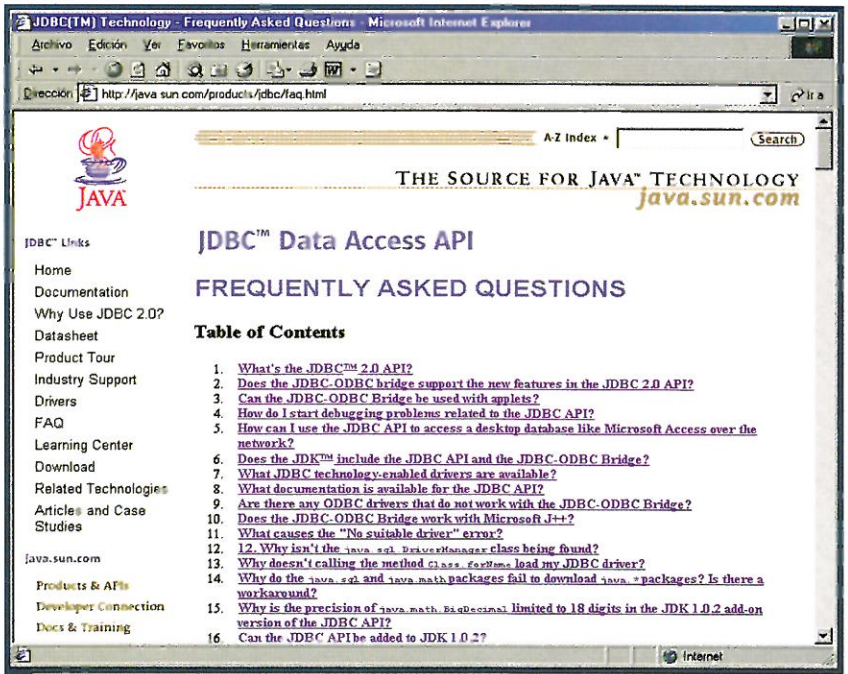
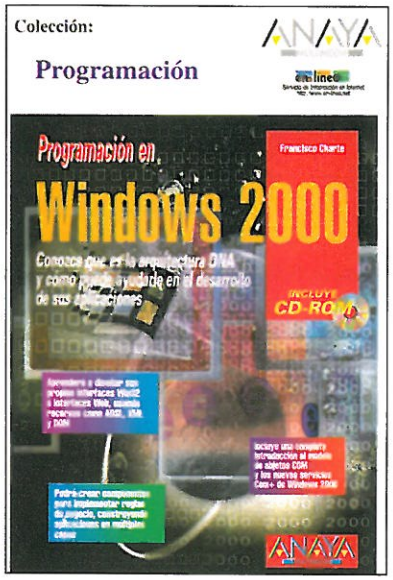


Figura 5.- El *FAQ* de *JBDC (java.sun.com/products/jdbc/faq.html)* es muy completo.

PROGRAMACIÓN EN WINDOWS 2000

Este es un libro dirigido a cualquier usuario avanzado que quiera aumentar sus conocimientos de programación en *Windows 2000*. En efecto, las principales novedades de *Windows 2000* las encontraremos en la arquitectura *Windows DNA*, que se nos ofrece como un modelo de construcción de soluciones más flexible y eficiente que el proporcionado por *Windows NT* o *9x*.



De forma sencilla y clara, *Programación en Windows 2000* nos guiará a través de *COM* y *Com +*, *OLE DB* y *ADO, XML* y *XSL, RDS*, y todos los demás nuevos servicios de componentes y directorios distribuidos que ofrece la última versión de *Windows 2000*.

El libro se estructura en 20 contenidos temáticos, desde *Windows Distributed InterNet Application Architecture (DNA)*, al *Windows Scripting Host*, pasando por el diseño de interfaces *Win32 y WEB*.

La obra viene acompañada de dos *CD-ROM* con ejemplos, y la herramienta de *Microsoft Windows 2000 Developer´s Readiness Kit*.

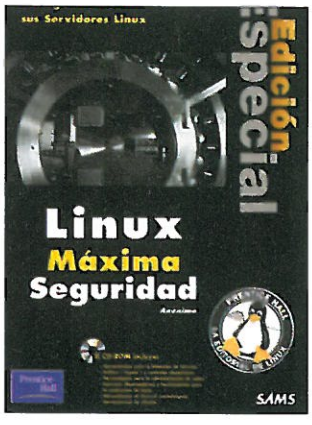
Editorial: Anaya Multimedia
Nº de páginas: 704
Nivel: Medio y avanzado

Autor: Francisco Charte
Idioma: Español
Precio: 5.995 Ptas. (I.V.A inc.)

LINUX MÁXIMA SEGURIDAD

Interesante libro dedicado a la protección de servidores y estaciones de trabajo *Linux* contra intrusiones críticas para la integridad del sistema.

El autor, según afirma el editor, es un "experimentado *Hacker* informático condenado por varios delitos económicos tras desarrollar una técnica para burlar la seguridad del sistema de cajeros", y felizmente recuperado como escritor y consejero de seguridad de sistemas informáticos.



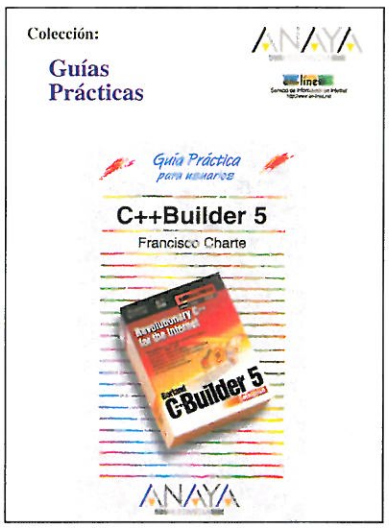
Y como "no hay mejor cuña que la de la misma madera", hay que suponer que el anónimo autor del libro estará en condiciones de darnos los mejores consejos para asegurar nuestro sistema contra agresiones externas, desde la seguridad física y la mejor instalación, a la seguridad en el correo electrónico y diseño de Web fiable, pasando por la detección de intrusiones y la recuperación de información corrupta o dañada. Esta publicación desvelará además, el modo en que *hackers, sniffers y spoofers* se introducen en el sistema, sus herramientas preferidas, y los trucos para neutralizarlos, utilizando tecnologías como *SSL* y *MDS*.

Incluye un *CD-ROM* con herramientas de administración, detección, cifrado, evaluación de *host*, y *firewalls*.

Editorial: Prentice Hall
Nº de páginas: 780
Nivel: Avanzado

Autor: Anónimo
Idioma: Español
Precio: 7.995 Ptas. (I.V.A inc.)

C++ BUILDER 5



Anaya nos presenta esta *Guía Práctica* a una de las mejores herramientas de desarrollo en lenguaje *C++*, ideal tanto para los principiantes en la programación, como para los programadores más experimentados.

En este manual, conoceremos los fundamentos del lenguaje *C++*, y aprenderemos a crear interfaces personalizadas, gestionar proyectos y excepciones, manipular componentes, trabajar con bases de datos, y generar informes y aplicaciones distribuidas en *CORBA*.

Después de leer el artículo de este número dedicado a *Builder 5*, seguro que te va a interesar este libro, en el que encontrarás una excelente referencia para llegar a ser un experto en el desarrollo de 32 bits para Windows.

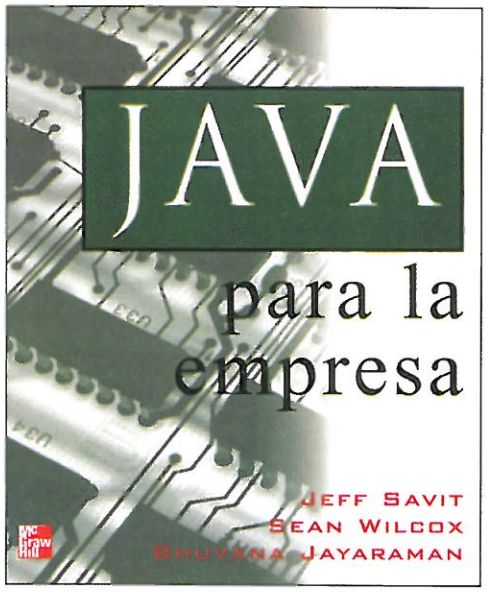
Te recordamos que en el *CD* de la revista encontrarás el *Builder 5 Enterprise*, y desde la página de *Borland* puedes descargar en tu máquina la versión *C++ Builder Compiler 5.5*. Es plenamente operativa, y no te costará un duro.

Editorial: Anaya
Número de páginas: 320
Nivel: Medio

Autor: Francisco Charte Ojeda
Idioma: Español
Precio: 1.700 Ptas. (I.V.A. inc.)

JAVA PARA LA EMPRESA

El perfecto conocimiento y dominio del lenguaje *Java*, se ha convertido en condición indispensable para la labor de los desarrolladores de aplicaciones y los gerentes de sistemas de información, dado el gran dinamismo y continuo desarrollo experimentado por las tecnologías basadas en *Java*.



Este libro viene a ser una guía práctica dedicada al aprovechamiento de una de las plataformas preferidas en los sistemas corporativos de cliente-servidor para sacar el mayor rendimiento de las capacidades del *World Wide Web*. Sus autores, experimentados profesionales de la aplicación *WEB*, abordan la materia desde los mismos fundamentos de *Java*, hasta su uso en la computación empresarial, según los requerimientos de los proveedores de sistema.

En estas páginas, aprenderemos todo lo necesario sobre las aplicaciones empresariales basadas en *WWW*, *Intranet* e *Internet*, además del empleo de *Java* fuera del *WEB*, como la última evolución de los lenguajes *C*, a los que *Java* puede llegar a desplazar en un futuro no muy lejano.

Editorial: McGraw Hill Interamericana
Nº de páginas: 350
Nivel: Avanzado

Autores: Savit, Wilcox y Jayaraman
Idioma: Español
Precio: 3.800 Ptas. (I.V.A. inc.)

Dudas técnicas

Estas páginas están dedicadas a resolver cualquier duda o problema que nos queráis plantear. Podéis dirigir vuestras preguntas a SÓLO PROGRAMADORES mandando un E-mail a nuestra dirección: solop@virtualsw.es

PREGUNTA

Hola, Programadores:

Antes de nada, quisiera dar la enhorabuena a todos los que formáis la revista ya que es la mejor. Estoy realizando unas páginas para ponerlas en mi servidor, y quisiera saber si es posible averiguar con *Javascript* cuál es el ancho de la ventana del navegador en la que se está viendo una determinada página, y si en función de ese ancho puedo determinar cosas tales como el ancho de una tabla.

Otra de las cosas que me gustaría incluir en mis páginas sería un calendario del mes actual donde se iluminara la semana actual. He empezado con algunas pruebas pero no saco nada en claro ¿Puedo hacer esto con *Javascript*?, ¿Cómo?

Espero vuestra respuesta. Muchas gracias.

Benito Alarcón.

RESPUESTA

Estimado lector:

Respondiendo a tus preguntas, debemos decirte que sí es posible calcular la anchura y la altura de la ventana del navegador. Esto depende del tipo de navegador que se utilice, como verás en la Tabla 1, que te mostramos a continuación.

TABLA 1 .Equivalencia de dimensiones de ventana según navegador

	Netscape Communicator	Internet Explorer
Anchura	window.innerWidth	Document.body.clientWidth
Altura	window.innerHeight	Document.body.clientHeight

Si el propósito es fijar el tamaño de otros elementos dentro de la página a partir del tamaño de la ventana del navegador disponible, lo más cómodo es añadir un pequeño *script* al principio de la página, que guarde en dos variables globales los valores relativos a la anchura y a la altura, con independencia del navegador utilizado.

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!-
var anchura = 0;
var altura = 0;
if (document.layers) {
  anchura = window.innerWidth;
  altura = window.innerHeight;
} else {
  anchura = document.body.client-
    Width;
  altura = document.body.client-
    Height;
}
//->
</SCRIPT>
</HEAD>
<BODY>
...
</BODY>
</HTML>
```

Ahora podríamos hacer que una imagen tomara por ejemplo los siguientes valores de anchura y altura:

```
<SCRIPT LANGUAGE="JavaScript">
<!-
document.write('<IMG SRC="i.gif"
    NAME="mi_img" WIDTH="' +
    (anchura/2) + '" HEIGTH="' +
    (altura/3) + '">');
//->
</SCRIPT>
```

El ejemplo antes citado no funciona sin embargo para *Internet Explorer,* porque para este navegador la anchura y la altura son

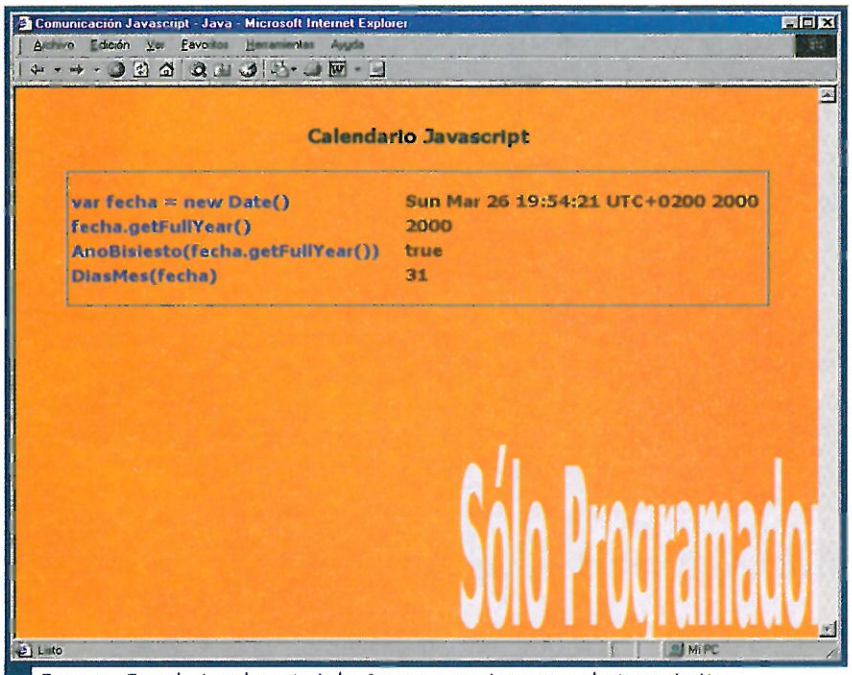


Figura 1. - Ejemplo de utilización de las funciones que determinan el número de días del mes en curso.

valores que dependen del objeto *document*, y éste existe únicamente cuando toda la página ha sido cargada. El *script*, tal y como está situado, se ejecuta mucho antes de que se empiece a analizar el cuerpo de la página, delimitado por la etiqueta *<BODY>*.

Por lo tanto, para que lo anterior funcionase, sería preciso crear una nueva función que se ejecutara al cargarse la página en el navegador.

```
<BODY onload="Inicializar()">
```

Esta función se mostraría con el siguiente código:

```
function Inicializar() {
  if (!document.layers) {

    document.images["mi_img"].wit
    dh = anchura;

    document.images["mi_img"].alt
    ura = altura;
  }
}
```

Con el código anterior, *Internet Explorer* mostraría también la imagen en función del tamaño de la ventana.

Para contestarte a la segunda de tus preguntas, te señalamos que no resulta demasiado difícil desarrollar un calendario dinámico con *Javascript*. Aunque realizar todo el *script* excede con mucho esta sección, sí que podemos darte algunas ideas que te sirvan para acercarte a tu propósito. En primer lugar, tienes que calcular la fecha actual:

```
var fecha = new Date();
```

Para recorrer todos los días del mes, es preciso conocer el número total del mismo. También se necesita saber si el año es bisiesto o no. Las siguientes funciones resuelven ambos problemas:

```
function AnoBisiesto(ano) {
  return (ano % 4 == 0 && (ano %
    100 != 0 || ano % 400 ==
    0));
}
```

```
function DiasMes(fecha) {
  var mes = fecha.getMonth() + 1;
  var ano = fecha.getYear();
  if ((mes==4)||(mes==6)||
      (mes==9)||(mes==11)) return
    30;
  if ((mes==1)||(mes==3)||
      (mes==5)||(mes==7)||
      (mes==9)||(mes==10)||
      (mes==12)) return 31;
  if (AnoBisiesto(ano)) {
    return 29;
  } else {
    return 28;
  }
}
```

De esta forma, resulta muy sencillo elaborar un calendario a partir de la fecha actual.

PREGUNTA

Un saludo a todos los amigos de SOLO PROGRAMADORES:

Estoy haciendo un pequeño programa en *Perl* que recoge datos procedentes de un fichero de texto y los procesa. Lo cierto es que no tengo mucha experiencia, y estoy bastante liado. He oído decir que con *Perl* se pueden utilizar *arrays* ,y he estado haciendo algunas pruebas pero sin demasiado éxito. ¿Cómo inicializo los *arrays*? ¿Cómo los puedo recorrer?

Muchas gracias.

Belén Pérez.

RESPUESTA

Estimada lectora:

En *Perl*, las variables que contienen arrays empiezan por el carácter (@). Ejemplos de *arrays* son los siguientes:

```
@array_vacio = ();
@array_numerico = (7, 8, 25, 6);
@array_cadenas = ("Burning",
```

Figura 2. - En *www.perl.com* podemos encontrar todo tipo de información relacionada con *Perl*.

```
"down", 'the', "house");
@array_mixto = ("Burning", 7,
     "down", 8, 'the', 25,
     "house", 6);
```

Los *arrays*, también pueden crearse a partir de otros pequeños *arrays*:

```
@mi_array1 = (5..10);
@mi_array2 = (1..5);
@mi_array = (@mi_array2,
     @mi_array1);
```

La primera de las expresiones crea un *array* con los elementos 5, 6, 7, 8, 9 y 10. La segunda, da lugar a un *array* con los elementos 1, 2, 3, 4 y 5. El resultado de concatenar los dos *arrays* anteriores sería otro con los elementos 1, 2, 3, 4, 5, 5, 6, 7, 8, 9 y 10.

Podemos acceder a los elementos individuales de un *array* de la siguiente forma:

```
@array_numerico = (1..5);
print $array_numerico[4];
```

Los elementos se numeran del *0* hasta *n-1*, donde *n* es el número de elementos que conforman el *array*.

También, pueden utilizarse índices negativos como así lo muestra el siguiente código:

```
@array_numerico = (1..5);
print $array_numerico[-1];
print $array_numerico[-2];
print $array_numerico[-3];
print $array_numerico[-4];
print $array_numerico[-5];
```

El ejemplo anterior permite mostrar los elementos 5, 4, 3, 2 y 1 del *array*.

Para conocer el número de elementos que conforman un *array*, basta asignarlo a una variable, tal y como sigue:

```
@array_numerico = (1..5);
$numero_elementos = @array_nume-
     rico;
```

Otra de las posibilidades más interesantes consiste en obtener *subarrays* a partir de un *array* dado:

```
@mi_array = ("uno", "dos",
     "tres", "cuatro");
@subarray = $mi_array[2,3];
print("\@subarray=@subarray\n");
```

El resultado de ejecutar la sentencia *print* anterior será:

```
@subarray=tres cuatro
```

Por último están los *arrays* asociativos, que son un poco diferentes a los que hemos visto hasta el momento. Un *array* asociativo es como uno normal en el que los índices pueden ser valores no numéricos.

```
%array_asociativo = ("Juan",
     "22", "Pepe", "24");
$array_asociativo{"Manuel"} =
```

La sentencia primera declara un *array* asociativo que equivale a una tabla del tipo (ver Tabla 2):

TABLA 2

Juan	**22**
Pepe	**24**

Al ejecutar la segunda de las sentencias, se añade un nuevo elemento a la tabla original ya que no había ninguno previamente con el nombre "Manuel". Para ver un ejemplo de esta acción, ver Tabla 3.

TABLA 3

Juan	**22**
Pepe	**24**
Manuel	**32**

Así, al ejecutar las siguientes sentencias *print*, se muestra por pantalla los números 22 y 32:

```
print $array_asociativo{"Juan"};
print $array_asociativo{"Manuel"};
```

Consultoría • Mantenimiento • Distribución • Formación • Programación a medida • Certificación • Material didáctico • Cursos para empresas

Universe Linux™

La alternativa que su empresa estaba esperando

Linux está en boca de todos: potencia, versatilidad, seguridad con mayúsculas, economía. Y todos se hacen las mismas preguntas. ¿Debo cambiar a **Linux**? ¿Es cierto que **Linux** es más fiable en la empresa que Windows NT, 98 o Windows 2000? ¿Es cierto que **Linux** puede convivir con Windows intercambiando información de forma transparente? ¿Puedo ahorrar con soluciones basadas en **Linux** hasta el 60% del precio en sus programas equivalentes de Microsoft? ¿Por qué más del 65% de los servidores de Internet corren bajo **Linux**? ¿Quién me dará el soporte y la garantía necesarios?
La respuesta a todas las preguntas se llama ***Universe Linux***

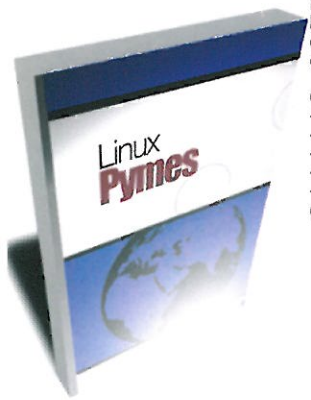
Universe Linux es la primera empresa de nuestro país dedicada exclusivamente a ofrecer soluciones basadas en **Linux** para usuarios y empresas: distribuciones **Linux**, utilidades, consultoría, instalaciones a medida, mantenimiento, formación... Todo lo que Vd. necesite y al mejor precio del mercado.

Puede que aún no sepa lo que ***Universe Linux*** puede hacer por Vd. Le invitamos a conocer nuestros productos con una demostración gratuita en nuestra sucursal de Madrid. Llámenos para concertar la demostración.
Infórmese sin compromiso llamando al **91-356 69 08** o visitando nuestra página web:

http://www.u-linux.com

Llame hoy mismo. *Por fin* en LINUX hay alguien que RESPONDE.

Linux PYMES
La distribución pensada para empresas
Pvp. recomendado 9.900 pts.



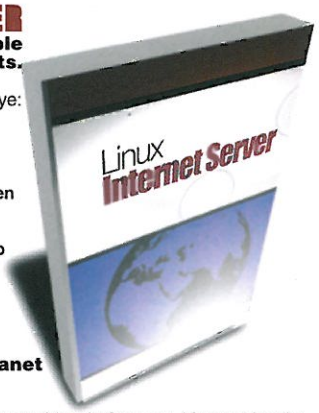
Por fin, la primera distribución creada para las PYMES: robusta, fiable, sencilla de instalar. Incluye: StarOffice, escritorio KDE, manual de instalación paso a paso y, como oferta de lanzamiento,GRATIS, el programa LINUX FAX SERVER, valorado en 14.900 pts. (*)

Características:
- Instalación en modo gráfico con pantallas de ayuda
- Configuración X completa antes de la instalación de paquetes
- Flexibilidad en los modos de instalación Server y Workstation
- Utilidades de automontaje
- XFree 3.3.5-3, Kernel 2.2.12-20, KDE 1.1.2, GTK, etc.

(*) Oferta válida hasta el 1 de abril del 2000.

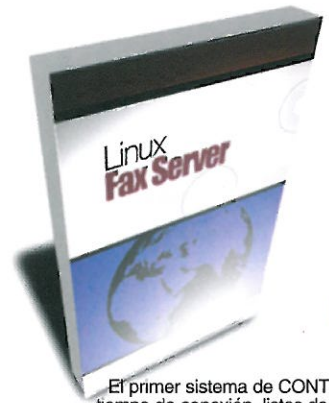
FORMACIÓN
Cursos de Linux todos los niveles,
Básico
Administrativo
Seminarios

Linux INTERNET SERVER
El servidor más fiable
Pvp. recomendado 49.950 pts.



El servidor de Internet pensado para su empresa incluye:
***Servidor de correo electrónico, páginas Web y FTP.**
***Servidor de proxy y caché.**
***Gestión y mantenimiento de listas de correo.**
***Alojamiento del Dominio y Dirección IP fija.**
***Gestión de usuarios y autentificación de los mismos en la entrada al sistema.**
***Control y visualización de usuarios conectados.**
***10 niveles de seguridad en el servidor de páginas Web local.**
***Programación de eventos de conexión con Internet.**
Ver demostración en http:// www.u-linux.com/demo

Linux FAX SERVER
Ahorre enviando todos sus fax por la intranet
Pvp. recomendado 24.900 pts.



Se acabaron las colas para enviar un fax. El primer servidor de fax para Linux selecciona entre los tramos horarios más económicos de su operador de comunicaciones y se adapta a ellos, economizando en cada envío. Evita desplazamientos dentro de la propia empresa ahorrando tiempo y molestias al utilizar la intranet para acceder al servidor. Crea las estadísticas necesarias para analizar los gastos por cada usuario. Economía y comodidad garantizadas en sus comunicaciones.

LINUX WATCHDOG CONTROLLER
El guardián de su empresa en Internet
Pvp. recomendado 69.950 pts. (*)

El primer sistema de CONTROL TOTAL sobre Internet en su empresa. Limitación en el tiempo de conexión, listas de direcciones, chat, en general los puertos más representativos del sistema. Estadísticas de conexión por usuario, por sites accedidas, por distribución del tiempo según operador. Limitaciones individuales o por grupos de usuarios. El sistema corta la conexión cuando detecta que el usuario ha sobrepasado los límites de sus privilegios y le envía un mensaje al supervisor. LINUX WATCHDOG CONTROLLER es una herramienta imprescindible en la empresa moderna para el control del uso de Internet. El 90% de los empleados de las empresas con acceso a Internet desde su puesto de trabajo navegan con fines personales: chats, compras, etc. Esta herramienta erradica el problema realizando auditorías individualizadas de cada usuario, ya que almacena direcciones accedidas, tiempo de estancia, información recibida y enviada, etc. LINUX WATCHDOG CONTROLLER es una utilidad imprescindible para los Administradores del Sistema y Jefes de Personal. Pruébelo sin compromiso.

(*) Hasta 10 puestos de trabajo. Consultar según necesidades superiores.
Los precios no incluyen IVA.

Universe Linux *le brinda las mejores opciones para* **hacer negocio:**

Linux Training Center
Formación a todos los niveles

Si dispone de una academia de informática Vd. puede tener el LINUX TRAINING CENTER oficial de su población. Universe Linux formará a su personal y le proporcionará todo el material didáctico necesario y la certificación LINUX TRAINING CENTER. Además, le incluiremos en nuestra Lista de Centros Autorizados en todas las acciones publicitarias promocionando su negocio. (*)
La más amplia gama de cursos para todos los niveles y necesidades en formación: Introducción a Linux, Administración de Sistemas, Instalación y configuración de Servidores web, Ofimática en Linux, etc.
Sistema de franquicias con número limitado de licencias por provincia.

(*) Más de 250.000 impactos cada mes en prensa especializada y general.

Linux Solution Provider
Una apuesta segura para emprendedores. Servicios en Linux

Si es Vd. socio o propietario de una empresa de Servicios Informáticos puede obtener para su provincia la representación en *exclusiva* de los productos **Universe Linux** y la certificación **LINUX SOLUTION PROVIDER** para su empresa de servicios, distribuyendo, instalando y dando soporte a la más amplia gama de productos Linux, con los mejores márgenes comerciales del mercado. Además, contará con el asesoramiento de nuestros técnicos y ayuda on line.
Un negocio de rentabilidad asegurada, sin cánones de entrada, compatible con su actividad y clientes habituales. No deje pasar la oportunidad e infórmese en el **91-356 69 08.**

Linux Developer
La mejor opción para desarrolladores

Universe Linux ofrece a los profesionales independientes del mundo de la programación la posibilidad de adquirir todos los conocimientos para trabajar en el mundo Linux. Incluye el material didáctico y el software necesario a un precio excepcional. Sólo para desarrolladores.

Y también: STAROFFICE, APPLIXWARE, ANTIVIRUS, bases de datos relacionales, servidores de ficheros, y mucho más.

Distribuidores
Universe Linux S.L. España ofrece las mejores condiciones para distribuidores, con descuentos desde el 25% hasta el 40% según tramos de compra. Infórmese sin compromiso en el teléfono: (91) 356 69 08.
Si desea solicitar una demostración gratuita de alguno de los programas, póngase en contacto con nosotros.

UNIVERSE LINUX, LINUX SOLUTION PROVIDER, LINUX TRAINING CENTER, LINUX PYMES, LINUX INTERNET SERVER, LINUX WATCHDOG CONTROLER, son marcas registradas de U-LINUX, S.A. España.

WINDOWS NT, WINDOWS 98, WINDOWS 2000, MICROSOFT son marcas registradas de MICROSOFT CORPORATION.

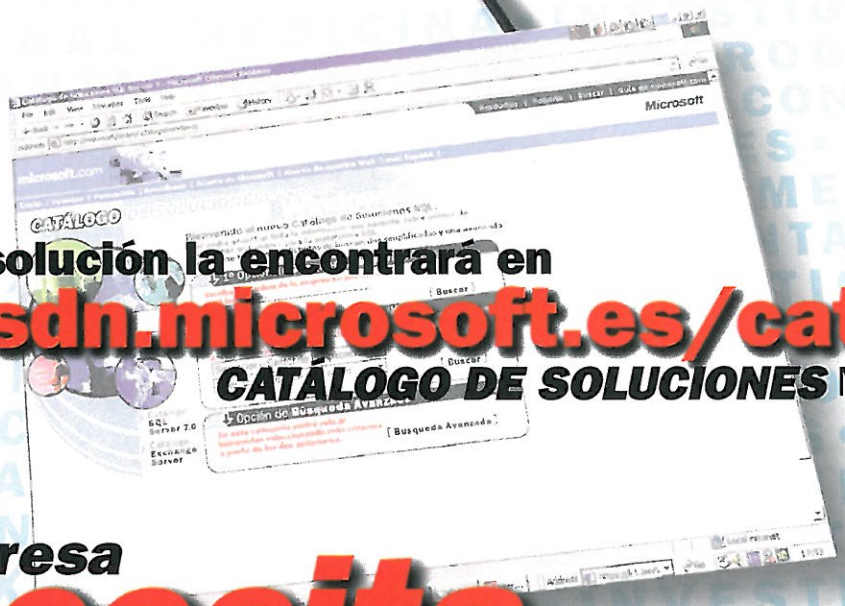
Universe Linux S.L. • C/ Conde de Vilches 21 • 28028 MADRID • Teléfono: (91) 356 69 08 • Fax: (91) 355 28 95 • http://www.u-linux.com

Su empresa

Tiene

Aplicaciones desarrolladas bajo Microsoft Windows

"Nuestra empresa ha desarrollado una aplicación. ¿Cuál será el sitio donde la puedan conocer el mayor número de clientes potenciales?"



La solución la encontrará en

msdn.microsoft.es/catalogo

CATÁLOGO DE SOLUCIONES Microsoft **Windows**

El lugar donde se encuentra el mayor número de aplicaciones desarrolladas bajo Microsoft Windows para cualquier tipo de empresas.

Su empresa

necesita

Aplicaciones de desarrollo bajo Microsoft Windows

"Buscamos una aplicación de gestión a medida para la actividad de nuestra empresa"

Las 50 primeras soluciones* que se incluyan en nuestro catálogo de soluciones, obtendrán un ratón IntelliMouse Explorer o un teclado Microsoft Internet Keyboard

* Solamente se obtendrá un regalo por empresa

Microsoft